The Clon Reference Manual

The Command-Line Options Nuker, version 1.0 beta 24 "Michael Brecker"

Didier Verna <didier@didierverna.net>

This manual was generated automatically by Declt 2.0 "Kathryn Janeway" on Mon Jul 13 18:46:17 2015 GMT+1.

Copyright © 2010-2012, 2015 Didier Verna

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the section entitled “Copying” is included exactly as in the original.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be translated as well.

Table of Contents

Copying **1**

1 Introduction **3**

2 Systems **5**

2.1 `net.didierverna.clon` 5
2.2 `net.didierverna.clon.termio` 5
2.3 `net.didierverna.clon.setup/termio` 6
2.4 `net.didierverna.clon.core` 7
2.5 `net.didierverna.clon.setup` 7

3 Modules **9**

3.1 `core/src/` 9
3.2 `core/src/options/` 9
3.3 `core/src/retrieval/` 10
3.4 `core/src/output/` 10

4 Files **11**

4.1 Lisp 11
4.1.1 `net.didierverna.clon.asd` 11
4.1.2 `net.didierverna.clon.termio.asd` 11
4.1.3 `net.didierverna.clon.core.asd` 11
4.1.4 `net.didierverna.clon.setup.asd` 11
4.1.5 `termio/sbcl/constants.lisp` 11
4.1.6 `termio/termio.lisp` 11
4.1.7 `core/meta.lisp` 11
4.1.8 `core/src/util.lisp` 12
4.1.9 `core/src/item.lisp` 12
4.1.10 `core/src/text.lisp` 13
4.1.11 `core/src/options/option.lisp` 13
4.1.12 `core/src/options/flag.lisp` 14
4.1.13 `core/src/options/valued.lisp` 14
4.1.14 `core/src/options/negatable.lisp` 15
4.1.15 `core/src/options/switch-base.lisp` 15
4.1.16 `core/src/options/switch.lisp` 15
4.1.17 `core/src/options/stropt.lisp` 16
4.1.18 `core/src/options/lispobj.lisp` 16
4.1.19 `core/src/options/path.lisp` 16
4.1.20 `core/src/options/enum-base.lisp` 17
4.1.21 `core/src/options/enum.lisp` 17
4.1.22 `core/src/options/xswitch.lisp` 17
4.1.23 `core/src/container.lisp` 18
4.1.24 `core/src/group.lisp` 18
4.1.25 `core/src/retrieval/cmdline.lisp` 18
4.1.26 `core/src/retrieval/environ.lisp` 19
4.1.27 `core/src/synopsis.lisp` 19

4.1.28 `core/src/output/face.lisp` ... 20
4.1.29 `core/src/output/sheet.lisp` ... 21
4.1.30 `core/src/context.lisp` ... 23
4.1.31 `setup/setup.lisp` ... 25

5 Packages ... 27
5.1 `net.didierverna.clon` ... 27
5.2 `net.didierverna.clon.setup` ... 39

6 Definitions ... 41
6.1 Exported definitions ... 41
6.1.1 Special variables ... 41
6.1.2 Macros ... 42
6.1.3 Functions ... 43
6.2 Internal definitions ... 49
6.2.1 Constants ... 49
6.2.2 Special variables ... 49
6.2.3 Macros ... 50
6.2.4 Functions ... 52
6.2.5 Generic functions ... 70
6.2.6 Conditions ... 91
6.2.7 Structures ... 97
6.2.8 Classes ... 99

Appendix A Indexes ... 117
A.1 Concepts ... 118
A.2 Functions ... 119
A.3 Variables ... 126
A.4 Data types ... 128

Copying

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THIS SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

1 Introduction

`Clon` is a library for managing command-line options in standalone Common-Lisp applications. It provides a unified option syntax with both short and long names, automatic completion of partial names and automatic retrieval/conversion of option arguments from the command-line, associated environment variables, fallback or default values. `Clon` comes with a set of extensible option types (switches, paths, strings *etc.*). `Clon` also provides automatic generation and formatting of help strings, with support for highlighting on `tty`'s through ISO/IEC 6429 SGR. This formatting is customizable through *themes*.

Depending on the target audience, `Clon` stands for either "The Command-Line Options Nuker" or "The Common-Lisp Options Nuker". `Clon` also has a recursive acronym: "`Clon` Likes Options Nuking", and a reverse one: "Never Omit to Link with `Clon`". Other possible expansions of the acronym are still being investigated.

This is the `Clon` reference manual, and as such, it is not meant to be read. It may help you find sleep in case of insomnia though. `Clon` comes with two human-readable manuals:

- the "end-user manual" (see *The Clon End-User Manual*) is for the `Clon` *end-user*, that is, the user of an application powered by `Clon`. It describes how to use the command-line of clonified[1] applications and how to customize `Clon`'s output. Everybody should read this manual first.
- the "user manual" (see *The Clon End-User Manual*) is for the `Clon` *user*, that is, the developer of a Common-Lisp application who wants to use `Clon` for command-line option management. It describes how to clonify your application and extend the library with your own option types.

[1] An application using `Clon` for its command-line option management is said to be *clonified*. It is also possible to say *clonfiscated*. However, we advise against using *clonistified*. The term *clonificated* is also considered bad style, and the use of *clonificationated* is strictly prohibited.

2 Systems

The main system appears first, followed by any subsystem dependency.

2.1 net.didierverna.clon

Long Name
The Command-Line Options Nuker

Author Didier Verna

Contact didier@didierverna.net

Home Page
http://www.lrde.epita.fr/~didier/software/lisp/clon.php

Source Control
https://github.com/didierverna/clon

License BSD

Description
Command-line options management for standalone Common Lisp applications

Long Description
Clon is a library for command-line options management. It is intended to ease the creation of standalone Common Lisp applications by providing a powerful and uniform command-line options interface. The most important features of Clon are the following.

- From the application programmer's point of view: centralized command-line options specification and management, including automatic generation of help strings, conversion from command-line / defaults / fallbacks / environment variables to application-level option values, global or on-demand option retrieval, and extensibility (the programmer can define his own option types).

- From the application user's point of view: uniform command-line option syntax across all Clon applications, customization of the help strings layout (with optional ISO6429 coloring on terminals that support it), automatic completion of abbreviated option names and short/long/pack syntax.

Dependencies
- [net.didierverna.clon.setup], page 7 (system)
- [net.didierverna.clon.core], page 7 (system)
- [net.didierverna.clon.termio], page 5 (system) (for feature net.didierverna.clon.termio)

Definition file
[net.didierverna.clon.asd], page 11 (Lisp file)

2.2 net.didierverna.clon.termio

Long Name
The Command-Line Options Nuker, termio library

Author Didier Verna

Contact `didier@didierverna.net`

Home Page
`http://www.lrde.epita.fr/~didier/software/lisp/clon.php`

Source Control
`https://github.com/didierverna/clon`

License BSD

Description
Clon's support for termio (tty geometry and fontification)

Long Description
Clon's termio library provides automatic detection of tty geometry and ISO6429 coloring on terminals that support it. For a more complete description of Clon, see the net.didierverna.clon system.

If Feature `net.didierverna.clon.termio`

Defsystem Dependencies
- [`net.didierverna.clon.setup/termio`], page 6 (system)
- required module `sb-grovel` (for feature `sbcl`)

Dependencies
- `sb-posix` (for feature `sbcl`)
- `cffi` (for feature `(and clisp net.didierverna.clon.termio)`)
- [`net.didierverna.clon.core`], page 7 (system)

Definition file
[`net.didierverna.clon.termio.asd`], page 11 (Lisp file)

Components
- [`sbcl/constants`], page 11 (Lisp file)
- [`termio`], page 11 (Lisp file)

2.3 `net.didierverna.clon.setup/termio`

Long Name
The Command-Line Options Nuker, termio setup

Author Didier Verna `<didier@didierverna.net>`

Contact `didier@didierverna.net`

Home Page
`http://www.lrde.epita.fr/~didier/software/lisp/clon.php`

Source Control
`https://github.com/didierverna/clon`

License BSD

Description
Clon's support for automatic configuration of termio support

Long Description
This is a virtual subsystem or Clon (no actual code). Its purpose is only to autodetect termio support and update Clon's preload configuration on load. For a more complete description of Clon, see the net.didierverna.clon system.

Dependency
[net.didierverna.clon.setup], page 7 (system)

Definition file
[net.didierverna.clon.setup.asd], page 11 (Lisp file)

2.4 net.didierverna.clon.core

Long Name
The Command-Line Options Nuker, core library

Author Didier Verna

Contact didier@didierverna.net

Home Page
http://www.lrde.epita.fr/~didier/software/lisp/clon.php

Source Control
https://github.com/didierverna/clon

License BSD

Description
Clon's basic, platform-independent functionality

Long Description
Clon's core library provides the platform/feature independent part. For a more complete description of Clon, see the net.didierverna.clon system.

Dependencies
- [net.didierverna.clon.setup], page 7 (system)
- required module sb-posix (for feature sbcl)

Definition file
[net.didierverna.clon.core.asd], page 11 (Lisp file)

Components
- [meta], page 11 (Lisp file)
- [src], page 9 (module)

2.5 net.didierverna.clon.setup

Long Name
The Command-Line Options Nuker, setup library

Author Didier Verna

Contact didier@didierverna.net

Home Page
http://www.lrde.epita.fr/~didier/software/lisp/clon.php

Source Control
https://github.com/didierverna/clon

License BSD

Description
Clon's preload setup library

Long Description

Clon's setup library provides support for various preload configuration parameters. For a more complete description of Clon, see the net.didierverna.clon system.

Definition file

[`net.didierverna.clon.setup.asd`], page 11 (Lisp file)

Component

[`setup`], page 25 (Lisp file)

3 Modules

Modules are listed depth-first from the system components tree.

3.1 core/src/

Dependency
[meta], page 11 (Lisp file)

Parent [net.didierverna.clon.core], page 7 (system)

Location core/src/

Components

- [util], page 12 (Lisp file)
- [item], page 12 (Lisp file)
- [text], page 13 (Lisp file)
- [options], page 9 (module)
- [container], page 18 (Lisp file)
- [group], page 18 (Lisp file)
- [retrieval], page 10 (module)
- [synopsis], page 19 (Lisp file)
- [output], page 10 (module)
- [context], page 23 (Lisp file)

3.2 core/src/options/

Dependency
[text], page 13 (Lisp file)

Parent [src], page 9 (module)

Location core/src/options/

Components

- [option], page 13 (Lisp file)
- [flag], page 14 (Lisp file)
- [valued], page 14 (Lisp file)
- [negatable], page 15 (Lisp file)
- [switch-base], page 15 (Lisp file)
- [switch], page 15 (Lisp file)
- [stropt], page 16 (Lisp file)
- [lispobj], page 16 (Lisp file)
- [path], page 16 (Lisp file)
- [enum-base], page 17 (Lisp file)
- [enum], page 17 (Lisp file)
- [xswitch], page 17 (Lisp file)

3.3 core/src/retrieval/

Dependency
[options], page 9 (module)

Parent [src], page 9 (module)

Location core/src/retrieval/

Components
- [cmdline], page 18 (Lisp file)
- [environ], page 19 (Lisp file)

3.4 core/src/output/

Dependencies
- [synopsis], page 19 (Lisp file)
- [retrieval], page 10 (module)

Parent [src], page 9 (module)

Location core/src/output/

Components
- [face], page 20 (Lisp file)
- [sheet], page 21 (Lisp file)

4 Files

Files are sorted by type and then listed depth-first from the systems components trees.

4.1 Lisp

4.1.1 net.didierverna.clon.asd

Location net.didierverna.clon.asd

4.1.2 net.didierverna.clon.termio.asd

Location termio/net.didierverna.clon.termio.asd

4.1.3 net.didierverna.clon.core.asd

Location core/net.didierverna.clon.core.asd

4.1.4 net.didierverna.clon.setup.asd

Location setup/net.didierverna.clon.setup.asd

4.1.5 termio/sbcl/constants.lisp

If Feature sbcl

Parent [net.didierverna.clon.termio], page 5 (system)

Location termio/sbcl/constants.lisp

4.1.6 termio/termio.lisp

Dependency
[sbcl/constants], page 11 (Lisp file)

Parent [net.didierverna.clon.termio], page 5 (system)

Location termio/termio.lisp

Internal Definitions

- [stream-ioctl-output-handle], page 87 (generic function)
- [stream-ioctl-output-handle], page 88 (method)
- [stream-ioctl-output-handle], page 88 (method)
- [stream-ioctl-output-handle], page 88 (method)
- [stream-ioctl-output-handle], page 88 (method)
- [stream-line-width], page 69 (function)

4.1.7 core/meta.lisp

Parent [net.didierverna.clon.core], page 7 (system)

Location core/meta.lisp

Packages [net.didierverna.clon], page 27

Exported Definitions
[nickname-package], page 48 (function)

Internal Definitions

- [*readtable*], page 50 (special variable)

- [clindent], page 53 (function)
- [defindent], page 50 (macro)
- [i-reader], page 58 (function)
- [in-readtable], page 51 (macro)
- [tilde-reader], page 69 (function)

4.1.8 core/src/util.lisp

Parent [src], page 9 (module)

Location core/src/util.lisp

Exported Definitions

- [cmdline], page 43 (function)
- [dump], page 42 (macro)
- [exit], page 44 (function)

Internal Definitions

- [*executablep*], page 49 (special variable)
- [abstract-class], page 99 (class)
- [accumulate], page 50 (macro)
- [beginning-of-string-p], page 53 (function)
- [closest-match], page 54 (function)
- [complete-string], page 54 (function)
- [copy-instance], page 74 (generic function)
- [copy-instance], page 75 (method)
- [declare-valid-superclass], page 50 (macro)
- [defabstract], page 50 (macro)
- [econd], page 51 (macro)
- [endpush], page 51 (macro)
- [getenv], page 57 (function)
- [home-directory], page 58 (function)
- [home-directory], page 93 (condition)
- [list-to-string], page 58 (function)
- [macosp], page 58 (function)
- [maybe-push], page 52 (macro)
- [putenv], page 64 (function)
- [remove-keys], page 65 (function)
- [replace-in-keys], page 52 (macro)
- [replace-key], page 65 (function)
- [replace-keys], page 66 (function)
- [select-keys], page 69 (function)

4.1.9 core/src/item.lisp

Dependency

[util], page 12 (Lisp file)

Parent [src], page 9 (module)

Location `core/src/item.lisp`

Internal Definitions

- [`help-spec`], page 77 (generic function)
- [`help-spec`], page 78 (method)
- [`hiddenp`], page 78 (method)
- [`item`], page 106 (class)
- [`traversedp`], page 89 (method)
- [`(setf traversedp)`], page 89 (method)
- [`untraverse`], page 90 (generic function)
- [`untraverse`], page 90 (method)

4.1.10 `core/src/text.lisp`

Dependency
[`item`], page 12 (Lisp file)

Parent [`src`], page 9 (module)

Location `core/src/text.lisp`

Exported Definitions
[`make-text`], page 47 (function)

Internal Definitions

- [`contents`], page 74 (method)
- [`help-spec`], page 77 (method)
- [`make-internal-text`], page 61 (function)
- [`text`], page 114 (class)
- [`untraverse`], page 90 (method)

4.1.11 `core/src/options/option.lisp`

Parent [`options`], page 9 (module)

Location `core/src/options/option.lisp`

Internal Definitions

- [`check-name-clash`], page 72 (generic function)
- [`check-name-clash`], page 72 (method)
- [`check-name-clash`], page 72 (method)
- [`check-name-clash`], page 72 (method)
- [`description`], page 75 (method)
- [`env-var`], page 75 (method)
- [`help-spec`], page 77 (method)
- [`long-name`], page 80 (method)
- [`match-option`], page 62 (function)
- [`negated-pack-char`], page 81 (generic function)
- [`negated-pack-char`], page 82 (method)
- [`option`], page 108 (class)
- [`option-abbreviation-distance`], page 62 (function)
- [`option-error`], page 95 (condition)

- [option-sticky-distance], page 82 (generic function)
- [option-sticky-distance], page 83 (method)
- [potential-pack-char], page 63 (function)
- [short-name], page 86 (method)
- [short-pack-char], page 87 (generic function)
- [short-pack-char], page 87 (method)
- [untraverse], page 90 (method)

4.1.12 core/src/options/flag.lisp

Dependency
[option], page 13 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/flag.lisp

Exported Definitions
[make-flag], page 45 (function)

Internal Definitions
- [flag], page 106 (class)
- [make-internal-flag], page 59 (function)

4.1.13 core/src/options/valued.lisp

Dependency
[option], page 13 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/valued.lisp

Internal Definitions
- [*item-names*], page 50 (special variable)
- [argument-name], page 70 (method)
- [argument-required-p], page 70 (method)
- [check], page 71 (generic function)
- [convert], page 74 (generic function)
- [default-value], page 75 (method)
- [defoption], page 51 (macro)
- [fallback-value], page 76 (method)
- [help-spec], page 77 (method)
- [invalid-argument], page 93 (condition)
- [invalid-value], page 95 (condition)
- [option-sticky-distance], page 83 (method)
- [read-argument], page 64 (function)
- [read-value], page 65 (function)
- [restartable-check], page 66 (function)
- [restartable-convert], page 67 (function)
- [short-pack-char], page 87 (method)
- [short-syntax-help-spec-prefix], page 87 (generic function)

- [short-syntax-help-spec-prefix], page 87 (method)
- [stringify], page 88 (generic function)
- [valued-option], page 114 (class)

4.1.14 core/src/options/negatable.lisp

Dependency
[valued], page 14 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/negatable.lisp

Internal Definitions
- [negatable], page 108 (class)
- [negated-pack-char], page 82 (method)
- [short-syntax-help-spec-prefix], page 87 (method)

4.1.15 core/src/options/switch-base.lisp

Dependency
[negatable], page 15 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/switch-base.lisp

Internal Definitions
- [argument-style], page 70 (method)
- [argument-styles], page 70 (method)
- [(setf argument-styles)], page 70 (method)
- [no-values], page 82 (method)
- [(setf no-values)], page 82 (method)
- [switch-base], page 112 (class)
- [yes-values], page 91 (method)
- [(setf yes-values)], page 91 (method)

4.1.16 core/src/options/switch.lisp

Dependency
[switch-base], page 15 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/switch.lisp

Exported Definitions
[make-switch], page 47 (function)

Internal Definitions
- [check], page 72 (method)
- [convert], page 74 (method)
- [make-internal-switch], page 61 (function)
- [stringify], page 88 (method)
- [switch], page 111 (class)

4.1.17 core/src/options/stropt.lisp

Dependency
[valued], page 14 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/stropt.lisp

Exported Definitions
[make-stropt], page 46 (function)

Internal Definitions
- [check], page 72 (method)
- [convert], page 74 (method)
- [make-internal-stropt], page 60 (function)
- [stringify], page 88 (method)
- [stropt], page 111 (class)

4.1.18 core/src/options/lispobj.lisp

Dependency
[valued], page 14 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/lispobj.lisp

Exported Definitions
[make-lispobj], page 45 (function)

Internal Definitions
- [check], page 71 (method)
- [convert], page 74 (method)
- [lispobj], page 107 (class)
- [make-internal-lispobj], page 59 (function)
- [stringify], page 88 (method)
- [typespec], page 90 (method)

4.1.19 core/src/options/path.lisp

Dependency
[valued], page 14 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/path.lisp

Exported Definitions
[make-path], page 46 (function)

Internal Definitions
- [check], page 71 (method)
- [convert], page 74 (method)
- [directory-pathname-p], page 55 (function)
- [make-internal-path], page 60 (function)
- [path], page 109 (class)

- [path-type], page 83 (method)
- [pathname-component-null-p], page 63 (function)
- [split-path], page 69 (function)
- [stringify], page 88 (method)

4.1.20 core/src/options/enum-base.lisp

Parent [options], page 9 (module)

Location core/src/options/enum-base.lisp

Internal Definitions

- [enum], page 75 (method)
- [enum-base], page 102 (class)

4.1.21 core/src/options/enum.lisp

Dependencies

- [valued], page 14 (Lisp file)
- [enum-base], page 17 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/enum.lisp

Exported Definitions

[make-enum], page 45 (function)

Internal Definitions

- [check], page 71 (method)
- [convert], page 74 (method)
- [enum], page 102 (class)
- [make-internal-enum], page 59 (function)
- [stringify], page 88 (method)

4.1.22 core/src/options/xswitch.lisp

Dependencies

- [valued], page 14 (Lisp file)
- [switch-base], page 15 (Lisp file)
- [enum-base], page 17 (Lisp file)

Parent [options], page 9 (module)

Location core/src/options/xswitch.lisp

Exported Definitions

[make-xswitch], page 48 (function)

Internal Definitions

- [check], page 71 (method)
- [convert], page 74 (method)
- [make-internal-xswitch], page 61 (function)
- [stringify], page 88 (method)
- [xswitch], page 116 (class)

4.1.23 core/src/container.lisp

Dependency
[options], page 9 (module)

Parent [src], page 9 (module)

Location core/src/container.lisp

Internal Definitions
- [check-name-clash], page 72 (method)
- [check-name-clash], page 72 (method)
- [check-name-clash], page 72 (method)
- [container], page 99 (class)
- [help-spec], page 77 (method)
- [items], page 79 (method)
- [untraverse], page 90 (method)

4.1.24 core/src/group.lisp

Dependency
[container], page 18 (Lisp file)

Parent [src], page 9 (module)

Location core/src/group.lisp

Exported Definitions
- [defgroup], page 42 (macro)
- [make-group], page 45 (function)

Internal Definitions
- [%defgroup], page 50 (macro)
- [group], page 106 (class)
- [header], page 77 (method)
- [help-spec], page 77 (method)

4.1.25 core/src/retrieval/cmdline.lisp

Parent [retrieval], page 10 (module)

Location core/src/retrieval/cmdline.lisp

Internal Definitions
- [argument-popable-p], page 53 (function)
- [cmdline-convert], page 54 (function)
- [cmdline-error], page 91 (condition)
- [cmdline-option-error], page 92 (condition)
- [invalid-cmdline-argument], page 94 (condition)
- [invalid-negated-syntax], page 94 (condition)
- [maybe-pop-argument], page 51 (macro)
- [missing-cmdline-argument], page 95 (condition)
- [option-call-p], page 62 (function)
- [restartable-cmdline-convert], page 66 (function)

- [restartable-invalid-negated-syntax-error], page 52 (macro)
- [restartable-spurious-cmdline-argument-error], page 52 (macro)
- [retrieve-from-long-call], page 85 (generic function)
- [retrieve-from-long-call], page 85 (method)
- [retrieve-from-long-call], page 85 (method)
- [retrieve-from-negated-call], page 85 (generic function)
- [retrieve-from-negated-call], page 85 (method)
- [retrieve-from-negated-call], page 85 (method)
- [retrieve-from-negated-call], page 85 (method)
- [retrieve-from-short-call], page 85 (generic function)
- [retrieve-from-short-call], page 85 (method)
- [retrieve-from-short-call], page 85 (method)
- [spurious-cmdline-argument], page 96 (condition)

4.1.26 core/src/retrieval/environ.lisp

Parent [retrieval], page 10 (module)

Location core/src/retrieval/environ.lisp

Internal Definitions

- [environment-convert], page 55 (function)
- [environment-error], page 92 (condition)
- [environmental-option-error], page 93 (condition)
- [invalid-environment-value], page 94 (condition)
- [read-env-val], page 65 (function)
- [restartable-environment-convert], page 67 (function)
- [retrieve-from-environment], page 84 (generic function)
- [retrieve-from-environment], page 84 (method)
- [retrieve-from-environment], page 84 (method)
- [retrieve-from-environment], page 84 (method)

4.1.27 core/src/synopsis.lisp

Dependency

[group], page 18 (Lisp file)

Parent [src], page 9 (module)

Location core/src/synopsis.lisp

Exported Definitions

- [*synopsis*], page 41 (special variable)
- [defsynopsis], page 42 (macro)
- [make-synopsis], page 47 (function)

Internal Definitions

- [clon-options-group], page 72 (method)
- [do-options], page 51 (macro)
- [help-spec], page 77 (method)
- [mapoptions], page 80 (generic function)

- [mapoptions], page 81 (method)
- [mapoptions], page 81 (method)
- [mapoptions], page 81 (method)
- [mapoptions], page 81 (method)
- [negated-pack], page 81 (method)
- [postfix], page 83 (method)
- [potential-pack], page 84 (method)
- [potential-pack-p], page 84 (generic function)
- [potential-pack-p], page 84 (method)
- [short-pack], page 87 (method)
- [synopsis], page 113 (class)

4.1.28 core/src/output/face.lisp

Parent [output], page 10 (module)

Location core/src/output/face.lisp

Internal Definitions

- [*highlight-properties*], page 50 (special variable)
- [add-subface], page 52 (function)
- [attach-face-tree], page 53 (function)
- [background], page 71 (method)
- [blink], page 71 (method)
- [bottom-padding], page 71 (method)
- [concealedp], page 74 (method)
- [crossed-out-p], page 75 (method)
- [face], page 102 (class)
- [face-highlight-property-set-p], page 56 (function)
- [face-highlight-property-value], page 56 (function)
- [foreground], page 76 (method)
- [framedp], page 76 (method)
- [intensity], page 78 (method)
- [inversep], page 79 (method)
- [italicp], page 79 (method)
- [item-separator], page 79 (method)
- [left-padding], page 80 (method)
- [make-face-tree], page 80 (generic function)
- [make-face-tree], page 80 (method)
- [make-face-tree], page 80 (method)
- [make-raw-face-tree], page 61 (function)
- [name], page 81 (method)
- [parent], page 83 (method)
- [parent-generation], page 63 (function)
- [right-padding], page 86 (method)
- [search-branch], page 67 (function)

- [search-face], page 68 (function)
- [subface], page 88 (generic function)
- [subface], page 88 (method)
- [subface], page 88 (method)
- [subfaces], page 89 (method)
- [top-padding], page 89 (method)
- [underline], page 90 (method)
- [visiblep], page 91 (method)

4.1.29 core/src/output/sheet.lisp

Dependency
[face], page 20 (Lisp file)

Parent [output], page 10 (module)

Location core/src/output/sheet.lisp

Internal Definitions

- [available-right-margin], page 53 (function)
- [close-frame], page 73 (generic function)
- [close-frame], page 73 (method)
- [close-frame], page 73 (method)
- [close-line], page 53 (function)
- [close-sface], page 53 (function)
- [column], page 73 (method)
- [(setf column)], page 73 (method)
- [copy-frame], page 55 (function)
- [copy-highlight-frame], page 55 (function)
- [copy-highlight-property-instance], page 55 (function)
- [current-frame], page 55 (function)
- [current-left-margin], page 55 (function)
- [current-right-margin], page 55 (function)
- [current-sface], page 55 (function)
- [find-sface], page 56 (function)
- [flush-sheet], page 56 (function)
- [frame], page 98 (structure)
- [frame-left-margin], page 56 (function)
- [(setf frame-left-margin)], page 56 (function)
- [frame-p], page 56 (function)
- [frame-right-margin], page 56 (function)
- [(setf frame-right-margin)], page 56 (function)
- [frame-sface], page 56 (function)
- [(setf frame-sface)], page 56 (function)
- [frames], page 76 (method)
- [(setf frames)], page 76 (method)
- [get-bottom-padding], page 77 (generic function)

- [get-bottom-padding], page 77 (method)
- [get-bottom-padding], page 77 (method)
- [get-top-padding], page 57 (function)
- [help-spec-items-will-print], page 57 (function)
- [help-spec-will-print], page 78 (generic function)
- [help-spec-will-print], page 78 (method)
- [help-spec-will-print], page 78 (method)
- [help-spec-will-print], page 78 (method)
- [highlight-frame], page 98 (structure)
- [highlight-frame-highlight-property-instances], page 57 (function)
- [(setf highlight-frame-highlight-property-instances)], page 57 (function)
- [highlight-frame-left-margin], page 57 (function)
- [(setf highlight-frame-left-margin)], page 57 (function)
- [highlight-frame-p], page 57 (function)
- [highlight-frame-right-margin], page 57 (function)
- [(setf highlight-frame-right-margin)], page 57 (function)
- [highlight-frame-sface], page 57 (function)
- [(setf highlight-frame-sface)], page 57 (function)
- [highlight-property-ecase], page 51 (macro)
- [highlight-property-instance], page 99 (structure)
- [highlight-property-instance-escape-sequence], page 57 (function)
- [highlight-property-instance-name], page 58 (function)
- [(setf highlight-property-instance-name)], page 58 (function)
- [highlight-property-instance-p], page 58 (function)
- [highlight-property-instance-value], page 58 (function)
- [(setf highlight-property-instance-value)], page 58 (function)
- [highlightp], page 78 (method)
- [line-width], page 80 (method)
- [make-frame], page 58 (function)
- [make-highlight-frame], page 59 (function)
- [make-highlight-property-instance], page 59 (function)
- [make-raw-sface], page 62 (function)
- [make-sheet], page 62 (function)
- [map-frames], page 51 (macro)
- [open-frame], page 82 (generic function)
- [open-frame], page 82 (method)
- [open-frame], page 82 (method)
- [open-line], page 62 (function)
- [open-next-line], page 62 (function)
- [open-sface], page 62 (function)
- [output-stream], page 83 (method)
- [pop-frame], page 63 (function)

- [princ-char], page 63 (function)
- [princ-highlight-property-instances], page 63 (function)
- [princ-spaces], page 63 (function)
- [princ-string], page 63 (function)
- [print-faced-help-spec], page 64 (function)
- [print-help], page 64 (function)
- [print-help-spec], page 84 (generic function)
- [print-help-spec], page 84 (method)
- [print-help-spec], page 84 (method)
- [print-help-spec], page 84 (method)
- [print-help-spec], page 84 (method)
- [print-help-spec], page 84 (method)
- [print-string], page 64 (function)
- [push-frame], page 64 (function)
- [reach-column], page 64 (function)
- [read-sface-tree], page 65 (function)
- [safe-left-margin], page 67 (function)
- [safe-right-margin], page 67 (function)
- [sface], page 110 (class)
- [sface-tree], page 86 (method)
- [sheet], page 110 (class)
- [sibling], page 87 (method)
- [top-padding], page 89 (method)
- [top-padding], page 89 (method)
- [try-read-sface-tree], page 69 (function)
- [try-read-theme], page 69 (function)

4.1.30 core/src/context.lisp

Dependency

[output], page 10 (module)

Parent [src], page 9 (module)

Location core/src/context.lisp

Exported Definitions

- [*context*], page 41 (special variable)
- [cmdline-options-p], page 43 (function)
- [cmdline-p], page 43 (function)
- [do-cmdline-options], page 42 (macro)
- [getopt], page 44 (function)
- [getopt-cmdline], page 44 (function)
- [help], page 44 (function)
- [make-context], page 44 (function)
- [multiple-value-getopt-cmdline], page 43 (macro)
- [progname], page 48 (function)

- [remainder], page 48 (function)
- [with-context], page 43 (macro)

Internal Definitions

- [clon-options-group], page 72 (method)
- [cmdline-junk-error], page 91 (condition)
- [cmdline-option], page 97 (structure)
- [cmdline-option-name], page 54 (function)
- [(setf cmdline-option-name)], page 54 (function)
- [cmdline-option-option], page 54 (function)
- [(setf cmdline-option-option)], page 54 (function)
- [cmdline-option-p], page 54 (function)
- [cmdline-option-source], page 54 (function)
- [(setf cmdline-option-source)], page 54 (function)
- [cmdline-option-value], page 54 (function)
- [(setf cmdline-option-value)], page 54 (function)
- [cmdline-options], page 73 (method)
- [(setf cmdline-options)], page 73 (method)
- [context], page 100 (class)
- [copy-cmdline-option], page 55 (function)
- [error-handler], page 76 (method)
- [exit-abnormally], page 56 (function)
- [highlight], page 78 (method)
- [invalid-negated-equal-syntax], page 94 (condition)
- [invalid-short-equal-syntax], page 95 (condition)
- [line-width], page 80 (method)
- [make-cmdline-option], page 58 (function)
- [mapoptions], page 81 (method)
- [negated-pack], page 81 (method)
- [postfix], page 83 (method)
- [potential-pack-p], page 84 (method)
- [print-error], page 64 (function)
- [read-call], page 65 (function)
- [read-long-name], page 65 (function)
- [restart-on-error], page 66 (function)
- [restartable-cmdline-junk-error], page 66 (function)
- [search-option], page 68 (function)
- [search-option-by-abbreviation], page 68 (function)
- [search-option-by-name], page 68 (function)
- [search-path], page 86 (method)
- [search-sticky-option], page 68 (function)
- [short-pack], page 87 (method)
- [synopsis], page 89 (method)
- [theme], page 89 (method)

- [unknown-cmdline-option-error], page 96 (condition)
- [unrecognized-negated-call-error], page 97 (condition)
- [unrecognized-short-call-error], page 97 (condition)
- [untraverse], page 90 (method)
- [with-context-error-handler], page 52 (macro)

4.1.31 setup/setup.lisp

Parent [net.didierverna.clon.setup], page 7 (system)

Location setup/setup.lisp

Packages [net.didierverna.clon.setup], page 39

Exported Definitions

- [*release-major-level*], page 41 (special variable)
- [*release-minor-level*], page 41 (special variable)
- [*release-name*], page 41 (special variable)
- [*release-status*], page 41 (special variable)
- [*release-status-level*], page 41 (special variable)
- [configuration], page 42 (special variable)
- [configuration], page 43 (function)
- [configure], page 44 (function)
- [setup-termio], page 48 (function)
- [version], page 48 (function)

Internal Definitions

- [%version], page 52 (function)
- [release-status-number], page 65 (function)
- [restrict-because], page 67 (function)

5 Packages

Packages are listed by definition order.

5.1 net.didierverna.clon

The Command-Line Options Nuker package.

Source [meta], page 11 (Lisp file)

Use List

- [net.didierverna.clon.setup], page 39
- common-lisp

Exported Definitions

- [*context*], page 41 (special variable)
- [*synopsis*], page 41 (special variable)
- [cmdline], page 43 (function)
- [cmdline-options-p], page 43 (function)
- [cmdline-p], page 43 (function)
- [defgroup], page 42 (macro)
- [defsynopsis], page 42 (macro)
- [do-cmdline-options], page 42 (macro)
- [dump], page 42 (macro)
- [exit], page 44 (function)
- [getopt], page 44 (function)
- [getopt-cmdline], page 44 (function)
- [help], page 44 (function)
- [make-context], page 44 (function)
- [make-enum], page 45 (function)
- [make-flag], page 45 (function)
- [make-group], page 45 (function)
- [make-lispobj], page 45 (function)
- [make-path], page 46 (function)
- [make-stropt], page 46 (function)
- [make-switch], page 47 (function)
- [make-synopsis], page 47 (function)
- [make-text], page 47 (function)
- [make-xswitch], page 48 (function)
- [multiple-value-getopt-cmdline], page 43 (macro)
- [nickname-package], page 48 (function)
- [progname], page 48 (function)
- [remainder], page 48 (function)
- [with-context], page 43 (macro)

Internal Definitions

- [%defgroup], page 50 (macro)

- [*executablep*], page 49 (special variable)
- [*highlight-properties*], page 50 (special variable)
- [*item-names*], page 50 (special variable)
- [*readtable*], page 50 (special variable)
- [+tiocgwinsz+], page 49 (constant)
- [abstract-class], page 99 (class)
- [accumulate], page 50 (macro)
- [add-subface], page 52 (function)
- [allocate-winsize], page 53 (function)
- [argument], page 70 (generic function)
- [argument], page 70 (method)
- [argument], page 70 (method)
- [argument], page 70 (method)
- [argument-name], page 70 (generic function)
- [argument-name], page 70 (method)
- [argument-popable-p], page 53 (function)
- [argument-required-p], page 70 (generic function)
- [argument-required-p], page 70 (method)
- [argument-style], page 70 (generic function)
- [argument-style], page 70 (method)
- [argument-styles], page 70 (generic function)
- [argument-styles], page 70 (method)
- [(setf argument-styles)], page 70 (method)
- [(setf argument-styles)], page 70 (generic function)
- [attach-face-tree], page 53 (function)
- [available-right-margin], page 53 (function)
- [background], page 71 (generic function)
- [background], page 71 (method)
- [beginning-of-string-p], page 53 (function)
- [blink], page 71 (generic function)
- [blink], page 71 (method)
- [bottom-padding], page 71 (generic function)
- [bottom-padding], page 71 (method)
- [check], page 71 (generic function)
- [check], page 71 (method)
- [check], page 71 (method)
- [check], page 71 (method)
- [check], page 71 (method)
- [check], page 72 (method)
- [check], page 72 (method)
- [check-name-clash], page 72 (generic function)
- [check-name-clash], page 72 (method)
- [check-name-clash], page 72 (method)

- [check-name-clash], page 72 (method)
- [check-name-clash], page 72 (method)
- [check-name-clash], page 72 (method)
- [check-name-clash], page 72 (method)
- [clindent], page 53 (function)
- [clon-options-group], page 72 (generic function)
- [clon-options-group], page 72 (method)
- [clon-options-group], page 72 (method)
- [close-frame], page 73 (generic function)
- [close-frame], page 73 (method)
- [close-frame], page 73 (method)
- [close-line], page 53 (function)
- [close-sface], page 53 (function)
- [closest-match], page 54 (function)
- [cmdline-convert], page 54 (function)
- [cmdline-error], page 91 (condition)
- [cmdline-junk-error], page 91 (condition)
- [cmdline-option], page 97 (structure)
- [cmdline-option-error], page 92 (condition)
- [cmdline-option-name], page 54 (function)
- [(setf cmdline-option-name)], page 54 (function)
- [cmdline-option-option], page 54 (function)
- [(setf cmdline-option-option)], page 54 (function)
- [cmdline-option-p], page 54 (function)
- [cmdline-option-source], page 54 (function)
- [(setf cmdline-option-source)], page 54 (function)
- [cmdline-option-value], page 54 (function)
- [(setf cmdline-option-value)], page 54 (function)
- [cmdline-options], page 73 (generic function)
- [cmdline-options], page 73 (method)
- [(setf cmdline-options)], page 73 (method)
- [(setf cmdline-options)], page 73 (generic function)
- [column], page 73 (generic function)
- [column], page 73 (method)
- [(setf column)], page 73 (method)
- [(setf column)], page 73 (generic function)
- [comment], page 73 (generic function)
- [comment], page 73 (method)
- [comment], page 73 (method)
- [complete-string], page 54 (function)
- [concealedp], page 73 (generic function)
- [concealedp], page 74 (method)
- [container], page 99 (class)

- [contents], page 74 (generic function)
- [contents], page 74 (method)
- [context], page 100 (class)
- [convert], page 74 (generic function)
- [convert], page 74 (method)
- [convert], page 74 (method)
- [convert], page 74 (method)
- [convert], page 74 (method)
- [convert], page 74 (method)
- [convert], page 74 (method)
- [copy-cmdline-option], page 55 (function)
- [copy-frame], page 55 (function)
- [copy-highlight-frame], page 55 (function)
- [copy-highlight-property-instance], page 55 (function)
- [copy-instance], page 74 (generic function)
- [copy-instance], page 75 (method)
- [crossed-out-p], page 75 (generic function)
- [crossed-out-p], page 75 (method)
- [current-frame], page 55 (function)
- [current-left-margin], page 55 (function)
- [current-right-margin], page 55 (function)
- [current-sface], page 55 (function)
- [declare-valid-superclass], page 50 (macro)
- [defabstract], page 50 (macro)
- [default-value], page 75 (generic function)
- [default-value], page 75 (method)
- [defindent], page 50 (macro)
- [defoption], page 51 (macro)
- [description], page 75 (generic function)
- [description], page 75 (method)
- [directory-pathname-p], page 55 (function)
- [do-options], page 51 (macro)
- [econd], page 51 (macro)
- [endpush], page 51 (macro)
- [enum], page 75 (generic function)
- [enum], page 75 (method)
- [enum], page 102 (class)
- [enum-base], page 102 (class)
- [env-val], page 75 (generic function)
- [env-val], page 75 (method)
- [env-var], page 75 (generic function)
- [env-var], page 75 (method)
- [env-var], page 75 (method)

- [environment-convert], page 55 (function)
- [environment-error], page 92 (condition)
- [environmental-option-error], page 93 (condition)
- [error-handler], page 76 (generic function)
- [error-handler], page 76 (method)
- [error-string], page 76 (generic function)
- [error-string], page 76 (method)
- [(setf error-string)], page 76 (method)
- [(setf error-string)], page 76 (generic function)
- [exit-abnormally], page 56 (function)
- [face], page 102 (class)
- [face-highlight-property-set-p], page 56 (function)
- [face-highlight-property-value], page 56 (function)
- [fallback-value], page 76 (generic function)
- [fallback-value], page 76 (method)
- [find-sface], page 56 (function)
- [flag], page 106 (class)
- [flush-sheet], page 56 (function)
- [foreground], page 76 (generic function)
- [foreground], page 76 (method)
- [frame], page 98 (structure)
- [frame-left-margin], page 56 (function)
- [(setf frame-left-margin)], page 56 (function)
- [frame-p], page 56 (function)
- [frame-right-margin], page 56 (function)
- [(setf frame-right-margin)], page 56 (function)
- [frame-sface], page 56 (function)
- [(setf frame-sface)], page 56 (function)
- [framedp], page 76 (generic function)
- [framedp], page 76 (method)
- [frames], page 76 (generic function)
- [frames], page 76 (method)
- [(setf frames)], page 76 (method)
- [(setf frames)], page 76 (generic function)
- [free-winsize], page 57 (function)
- [get-bottom-padding], page 77 (generic function)
- [get-bottom-padding], page 77 (method)
- [get-bottom-padding], page 77 (method)
- [get-top-padding], page 57 (function)
- [getenv], page 57 (function)
- [group], page 106 (class)
- [header], page 77 (generic function)
- [header], page 77 (method)

- [help-spec], page 77 (generic function)
- [help-spec], page 77 (method)
- [help-spec], page 77 (method)
- [help-spec], page 77 (method)
- [help-spec], page 77 (method)
- [help-spec], page 77 (method)
- [help-spec], page 77 (method)
- [help-spec], page 78 (method)
- [help-spec-items-will-print], page 57 (function)
- [help-spec-will-print], page 78 (generic function)
- [help-spec-will-print], page 78 (method)
- [help-spec-will-print], page 78 (method)
- [help-spec-will-print], page 78 (method)
- [hiddenp], page 78 (generic function)
- [hiddenp], page 78 (method)
- [highlight], page 78 (generic function)
- [highlight], page 78 (method)
- [highlight-frame], page 98 (structure)
- [highlight-frame-highlight-property-instances], page 57 (function)
- [(setf highlight-frame-highlight-property-instances)], page 57 (function)
- [highlight-frame-left-margin], page 57 (function)
- [(setf highlight-frame-left-margin)], page 57 (function)
- [highlight-frame-p], page 57 (function)
- [highlight-frame-right-margin], page 57 (function)
- [(setf highlight-frame-right-margin)], page 57 (function)
- [highlight-frame-sface], page 57 (function)
- [(setf highlight-frame-sface)], page 57 (function)
- [highlight-property-ecase], page 51 (macro)
- [highlight-property-instance], page 99 (structure)
- [highlight-property-instance-escape-sequence], page 57 (function)
- [highlight-property-instance-name], page 58 (function)
- [(setf highlight-property-instance-name)], page 58 (function)
- [highlight-property-instance-p], page 58 (function)
- [highlight-property-instance-value], page 58 (function)
- [(setf highlight-property-instance-value)], page 58 (function)
- [highlightp], page 78 (generic function)
- [highlightp], page 78 (method)
- [home-directory], page 58 (function)
- [home-directory], page 93 (condition)
- [i-reader], page 58 (function)
- [in-readtable], page 51 (macro)
- [intensity], page 78 (generic function)

- [intensity], page 78 (method)
- [invalid-argument], page 93 (condition)
- [invalid-cmdline-argument], page 94 (condition)
- [invalid-environment-value], page 94 (condition)
- [invalid-negated-equal-syntax], page 94 (condition)
- [invalid-negated-syntax], page 94 (condition)
- [invalid-short-equal-syntax], page 95 (condition)
- [invalid-value], page 95 (condition)
- [inversep], page 79 (generic function)
- [inversep], page 79 (method)
- [italicp], page 79 (generic function)
- [italicp], page 79 (method)
- [item], page 79 (generic function)
- [item], page 79 (method)
- [item], page 106 (class)
- [item-separator], page 79 (generic function)
- [item-separator], page 79 (method)
- [items], page 79 (generic function)
- [items], page 79 (method)
- [junk], page 79 (generic function)
- [junk], page 79 (method)
- [left-padding], page 79 (generic function)
- [left-padding], page 80 (method)
- [line-width], page 80 (generic function)
- [line-width], page 80 (method)
- [line-width], page 80 (method)
- [lispobj], page 107 (class)
- [list-to-string], page 58 (function)
- [long-name], page 80 (generic function)
- [long-name], page 80 (method)
- [macosp], page 58 (function)
- [make-cmdline-option], page 58 (function)
- [make-face-tree], page 80 (generic function)
- [make-face-tree], page 80 (method)
- [make-face-tree], page 80 (method)
- [make-frame], page 58 (function)
- [make-highlight-frame], page 59 (function)
- [make-highlight-property-instance], page 59 (function)
- [make-internal-enum], page 59 (function)
- [make-internal-flag], page 59 (function)
- [make-internal-lispobj], page 59 (function)
- [make-internal-path], page 60 (function)
- [make-internal-stropt], page 60 (function)

- [make-internal-switch], page 61 (function)
- [make-internal-text], page 61 (function)
- [make-internal-xswitch], page 61 (function)
- [make-raw-face-tree], page 61 (function)
- [make-raw-sface], page 62 (function)
- [make-sheet], page 62 (function)
- [map-frames], page 51 (macro)
- [mapoptions], page 80 (generic function)
- [mapoptions], page 81 (method)
- [mapoptions], page 81 (method)
- [mapoptions], page 81 (method)
- [mapoptions], page 81 (method)
- [mapoptions], page 81 (method)
- [match-option], page 62 (function)
- [maybe-pop-argument], page 51 (macro)
- [maybe-push], page 52 (macro)
- [missing-cmdline-argument], page 95 (condition)
- [name], page 81 (generic function)
- [name], page 81 (method)
- [name], page 81 (method)
- [name], page 81 (method)
- [negatable], page 108 (class)
- [negated-call], page 81 (generic function)
- [negated-call], page 81 (method)
- [negated-pack], page 81 (generic function)
- [negated-pack], page 81 (method)
- [negated-pack], page 81 (method)
- [negated-pack-char], page 81 (generic function)
- [negated-pack-char], page 82 (method)
- [negated-pack-char], page 82 (method)
- [no-values], page 82 (generic function)
- [no-values], page 82 (method)
- [(setf no-values)], page 82 (method)
- [(setf no-values)], page 82 (generic function)
- [offset-of-winsize-ws-col], page 49 (constant)
- [offset-of-winsize-ws-row], page 49 (constant)
- [offset-of-winsize-ws-xpixel], page 49 (constant)
- [offset-of-winsize-ws-ypixel], page 49 (constant)
- [open-frame], page 82 (generic function)
- [open-frame], page 82 (method)
- [open-frame], page 82 (method)
- [open-line], page 62 (function)
- [open-next-line], page 62 (function)

- [open-sface], page 62 (function)
- [option], page 82 (generic function)
- [option], page 82 (method)
- [option], page 108 (class)
- [option-abbreviation-distance], page 62 (function)
- [option-call-p], page 62 (function)
- [option-error], page 95 (condition)
- [option-sticky-distance], page 82 (generic function)
- [option-sticky-distance], page 83 (method)
- [option-sticky-distance], page 83 (method)
- [output-stream], page 83 (generic function)
- [output-stream], page 83 (method)
- [parent], page 83 (generic function)
- [parent], page 83 (method)
- [parent-generation], page 63 (function)
- [path], page 109 (class)
- [path-type], page 83 (generic function)
- [path-type], page 83 (method)
- [pathname-component-null-p], page 63 (function)
- [pop-frame], page 63 (function)
- [postfix], page 83 (generic function)
- [postfix], page 83 (method)
- [postfix], page 83 (method)
- [potential-pack], page 83 (generic function)
- [potential-pack], page 84 (method)
- [potential-pack-char], page 63 (function)
- [potential-pack-p], page 84 (generic function)
- [potential-pack-p], page 84 (method)
- [potential-pack-p], page 84 (method)
- [princ-char], page 63 (function)
- [princ-highlight-property-instances], page 63 (function)
- [princ-spaces], page 63 (function)
- [princ-string], page 63 (function)
- [print-error], page 64 (function)
- [print-faced-help-spec], page 64 (function)
- [print-help], page 64 (function)
- [print-help-spec], page 84 (generic function)
- [print-help-spec], page 84 (method)
- [print-help-spec], page 84 (method)
- [print-help-spec], page 84 (method)
- [print-help-spec], page 84 (method)
- [print-help-spec], page 84 (method)
- [print-string], page 64 (function)

- [push-frame], page 64 (function)
- [putenv], page 64 (function)
- [reach-column], page 64 (function)
- [read-argument], page 64 (function)
- [read-call], page 65 (function)
- [read-env-val], page 65 (function)
- [read-long-name], page 65 (function)
- [read-sface-tree], page 65 (function)
- [read-value], page 65 (function)
- [remove-keys], page 65 (function)
- [replace-in-keys], page 52 (macro)
- [replace-key], page 65 (function)
- [replace-keys], page 66 (function)
- [restart-on-error], page 66 (function)
- [restartable-check], page 66 (function)
- [restartable-cmdline-convert], page 66 (function)
- [restartable-cmdline-junk-error], page 66 (function)
- [restartable-convert], page 67 (function)
- [restartable-environment-convert], page 67 (function)
- [restartable-invalid-negated-syntax-error], page 52 (macro)
- [restartable-spurious-cmdline-argument-error], page 52 (macro)
- [retrieve-from-environment], page 84 (generic function)
- [retrieve-from-environment], page 84 (method)
- [retrieve-from-environment], page 84 (method)
- [retrieve-from-environment], page 84 (method)
- [retrieve-from-long-call], page 85 (generic function)
- [retrieve-from-long-call], page 85 (method)
- [retrieve-from-long-call], page 85 (method)
- [retrieve-from-negated-call], page 85 (generic function)
- [retrieve-from-negated-call], page 85 (method)
- [retrieve-from-negated-call], page 85 (method)
- [retrieve-from-negated-call], page 85 (method)
- [retrieve-from-short-call], page 85 (generic function)
- [retrieve-from-short-call], page 85 (method)
- [retrieve-from-short-call], page 85 (method)
- [right-padding], page 86 (generic function)
- [right-padding], page 86 (method)
- [safe-left-margin], page 67 (function)
- [safe-right-margin], page 67 (function)
- [search-branch], page 67 (function)
- [search-face], page 68 (function)
- [search-option], page 68 (function)
- [search-option-by-abbreviation], page 68 (function)

- [search-option-by-name], page 68 (function)
- [search-path], page 86 (generic function)
- [search-path], page 86 (method)
- [search-sticky-option], page 68 (function)
- [select-keys], page 69 (function)
- [sface], page 110 (class)
- [sface-tree], page 86 (generic function)
- [sface-tree], page 86 (method)
- [sheet], page 110 (class)
- [short-call], page 86 (generic function)
- [short-call], page 86 (method)
- [short-name], page 86 (generic function)
- [short-name], page 86 (method)
- [short-pack], page 86 (generic function)
- [short-pack], page 87 (method)
- [short-pack], page 87 (method)
- [short-pack-char], page 87 (generic function)
- [short-pack-char], page 87 (method)
- [short-pack-char], page 87 (method)
- [short-syntax-help-spec-prefix], page 87 (generic function)
- [short-syntax-help-spec-prefix], page 87 (method)
- [short-syntax-help-spec-prefix], page 87 (method)
- [sibling], page 87 (generic function)
- [sibling], page 87 (method)
- [size-of-winsize], page 49 (constant)
- [split-path], page 69 (function)
- [spurious-cmdline-argument], page 96 (condition)
- [stream-ioctl-output-handle], page 87 (generic function)
- [stream-ioctl-output-handle], page 88 (method)
- [stream-ioctl-output-handle], page 88 (method)
- [stream-ioctl-output-handle], page 88 (method)
- [stream-ioctl-output-handle], page 88 (method)
- [stream-line-width], page 69 (function)
- [stringify], page 88 (generic function)
- [stringify], page 88 (method)
- [stringify], page 88 (method)
- [stringify], page 88 (method)
- [stringify], page 88 (method)
- [stringify], page 88 (method)
- [stringify], page 88 (method)
- [stropt], page 111 (class)
- [subface], page 88 (generic function)
- [subface], page 88 (method)

- [subface], page 88 (method)
- [subfaces], page 89 (generic function)
- [subfaces], page 89 (method)
- [switch], page 111 (class)
- [switch-base], page 112 (class)
- [synopsis], page 89 (generic function)
- [synopsis], page 89 (method)
- [synopsis], page 113 (class)
- [text], page 114 (class)
- [theme], page 89 (generic function)
- [theme], page 89 (method)
- [tilde-reader], page 69 (function)
- [top-padding], page 89 (generic function)
- [top-padding], page 89 (method)
- [top-padding], page 89 (method)
- [top-padding], page 89 (method)
- [traversedp], page 89 (generic function)
- [traversedp], page 89 (method)
- [(setf traversedp)], page 89 (method)
- [(setf traversedp)], page 89 (generic function)
- [try-read-sface-tree], page 69 (function)
- [try-read-theme], page 69 (function)
- [typespec], page 90 (generic function)
- [typespec], page 90 (method)
- [underline], page 90 (generic function)
- [underline], page 90 (method)
- [unknown-cmdline-option-error], page 96 (condition)
- [unrecognized-negated-call-error], page 97 (condition)
- [unrecognized-short-call-error], page 97 (condition)
- [untraverse], page 90 (generic function)
- [untraverse], page 90 (method)
- [untraverse], page 90 (method)
- [untraverse], page 90 (method)
- [untraverse], page 90 (method)
- [untraverse], page 90 (method)
- [value], page 90 (generic function)
- [value], page 90 (method)
- [valued-option], page 114 (class)
- [visiblep], page 90 (generic function)
- [visiblep], page 91 (method)
- [winsize-ws-col], page 69 (function)
- [(setf winsize-ws-col)], page 69 (function)
- [winsize-ws-row], page 69 (function)

- [(setf winsize-ws-row)], page 69 (function)
- [winsize-ws-xpixel], page 69 (function)
- [(setf winsize-ws-xpixel)], page 69 (function)
- [winsize-ws-ypixel], page 70 (function)
- [(setf winsize-ws-ypixel)], page 70 (function)
- [with-context-error-handler], page 52 (macro)
- [with-winsize], page 52 (macro)
- [xswitch], page 116 (class)
- [yes-values], page 91 (generic function)
- [yes-values], page 91 (method)
- [(setf yes-values)], page 91 (method)
- [(setf yes-values)], page 91 (generic function)

5.2 net.didierverna.clon.setup

The Command-Line Options Nuker, setup.

Source [setup], page 25 (Lisp file)

Use List common-lisp

Used By List

[net.didierverna.clon], page 27

Exported Definitions

- [*release-major-level*], page 41 (special variable)
- [*release-minor-level*], page 41 (special variable)
- [*release-name*], page 41 (special variable)
- [*release-status*], page 41 (special variable)
- [*release-status-level*], page 41 (special variable)
- [configuration], page 42 (special variable)
- [configuration], page 43 (function)
- [configure], page 44 (function)
- [setup-termio], page 48 (function)
- [version], page 48 (function)

Internal Definitions

- [%version], page 52 (function)
- [release-status-number], page 65 (function)
- [restrict-because], page 67 (function)

6 Definitions

Definitions are sorted by export status, category, package, and then by lexicographic order.

6.1 Exported definitions

6.1.1 Special variables

`*context*` [Special Variable]
The current context.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`*release-major-level*` [Special Variable]
The major level of this release.

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`*release-minor-level*` [Special Variable]
The minor level of this release.

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`*release-name*` [Special Variable]
The name of this release.

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`*release-status*` [Special Variable]
The status of this release.

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`*release-status-level*` [Special Variable]
The status level of this release.

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`*synopsis*` [Special Variable]
The current synopsis.

Package [`net.didierverna.clon`], page 27

Source [`synopsis`], page 19 (Lisp file)

`configuration` [Special Variable]

The Clon configuration settings.
This variable contains a property list of configuration options.
Current options are:
- :swank-eval-in-emacs (Boolean)
- :restricted (Boolean)
- :dump (Boolean)

See section A.1 of the user manual for more information.

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

6.1.2 Macros

`defgroup` (**&rest** *KEYS* **&key** *HEADER HIDDEN*) **&body** *FORMS* [Macro]

Define a new group.
KEYS are initargs to MAKE-GROUP (currently, only :header).
Each form in FORMS will be treated as a new :item.
The CAR of each form is the name of the operation to perform: TEXT, GROUP, or an option class name. The rest are the arguments to the MAKE-<OP> function or the DEFGROUP macro.

Package [`net.didierverna.clon`], page 27

Source [`group`], page 18 (Lisp file)

`defsynopsis` (**&rest** *KEYS* **&key** *POSTFIX MAKE-DEFAULT*) **&body** *FORMS* [Macro]

Define a new synopsis.

Package [`net.didierverna.clon`], page 27

Source [`synopsis`], page 19 (Lisp file)

`do-cmdline-options` (*OPTION NAME VALUE SOURCE* **&key** *CONTEXT*) **&body** *BODY* [Macro]

Evaluate BODY over all command-line options in CONTEXT.
OPTION, NAME and VALUE are bound to each option's object, name used on the command-line and retrieved value.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`dump` *NAME FUNCTION* **&rest** *ARGS* [Macro]

Dump a standalone executable named NAME starting with FUNCTION.
ARGS may be any arguments understood by the underlying implementation's dumping facility. They will simply be passed along. Note that DUMP already passes some such arguments. Some of them are critical for the dumping facility (e.g. :executable) and cannot be overridden. Some others, however, will be if you provide them as well (e.g. :load-init-file).

Since executable dumping is not available in all supported implementations, this function behaves differently in some cases, as described below.

- ECL doesn't create executables by dumping a Lisp image, but relies on having toplevel code to execute instead, so this macro simply expands to a call to FUNCTION. This also means

that ARGS is unused.
- ABCL can't dump executables at all because of the underlying Java implementation, so this macro expands to just (PROGN) but creates a Java class file with a main function that creates an interpreter, loads
the file in which this macro call appears and calls FUNCTION. This also means that ARGS is unused.

Package [net.didierverna.clon], page 27

Source [util], page 12 (Lisp file)

`multiple-value-getopt-cmdline` (*OPTION NAME VALUE SOURCE* **&key** *CONTEXT*) **&body** *BODY* [Macro]
Get the next command-line option in CONTEXT. and evaluate BODY.
OPTION, NAME and VALUE are bound to the values returned by GETOPT-CMDLINE. BODY is executed only if there is a next command-line option.

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

`with-context` *CONTEXT* **&body** *BODY* [Macro]
Execute BODY with *context* bound to CONTEXT.

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

6.1.3 Functions

`cmdline` [Function]
Get the current application's command-line.
This command-line is not supposed to contain any Lisp implementation specific option; only user-level ones. When a standalone executable is dumped, this is always the case. When used interactively, this depends on the underlying Lisp implementation. See appendix A.5 of the user manual for more information.

Package [net.didierverna.clon], page 27

Source [util], page 12 (Lisp file)

`cmdline-options-p` **&key** *CONTEXT* [Function]
Return T if CONTEXT has any unprocessed options left.

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

`cmdline-p` **&key** *CONTEXT* [Function]
Return T if CONTEXT has anything on its command-line.

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

`configuration` *KEY* [Function]
Return KEY's value in the current Clon configuration.

Package [net.didierverna.clon.setup], page 39

Source [setup], page 25 (Lisp file)

`configure` *KEY VALUE* [Function]
Set KEY to VALUE in the current Clon configuration.

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`exit` **&optional** *STATUS* [Function]
Quit the current application with STATUS.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`getopt` **&rest** *KEYS* **&key** *CONTEXT SHORT-NAME LONG-NAME OPTION* [Function]
Get an option's value in CONTEXT.
The option can be specified either by SHORT-NAME, LONG-NAME, or directly via an OPTION object.
Return two values:
- the retrieved value,
- the value's source.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`getopt-cmdline` **&key** *CONTEXT* [Function]
Get the next command-line option in CONTEXT.
When there is no next command-line option, return nil.
Otherwise, return four values:
- the option object,
- the option's name used on the command-line,
- the retrieved value,
- the value source.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`help` **&key** *CONTEXT ITEM OUTPUT-STREAM SEARCH-PATH THEME LINE-WIDTH HIGHLIGHT* [Function]
Print CONTEXT's help.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`make-context` **&rest** *KEYS* **&key** *SYNOPSIS CMDLINE MAKE-CURRENT* [Function]
Make a new context.
- SYNOPSIS is the program synopsis to use in that context.
It defaults to *SYNOPSIS*.
- CMDLINE is the argument list (strings) to process.
It defaults to a POSIX conformant argv.
- If MAKE-CURRENT, make the new context current.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`make-enum` **&rest** *KEYS* **&key** *SHORT-NAME LONG-NAME DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENUM ENV-VAR FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

Make a new enum option.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings.
It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENUM is the set of possible values.
- ENV-VAR is the option's associated environment variable.
It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [`net.didierverna.clon`], page 27

Source [`enum`], page 17 (Lisp file)

`make-flag` **&rest** *KEYS* **&key** *SHORT-NAME LONG-NAME DESCRIPTION ENV-VAR HIDDEN* [Function]

Make a new flag.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings.
It defaults to nil.
- ENV-VAR is the flag's associated environment variable.
It defaults to nil.
- When HIDDEN, the option doesn't appear in help strings.

Package [`net.didierverna.clon`], page 27

Source [`flag`], page 14 (Lisp file)

`make-group` **&rest** *KEYS* **&key** *HEADER ITEM HIDDEN* [Function]

Make a new group.

Package [`net.didierverna.clon`], page 27

Source [`group`], page 18 (Lisp file)

`make-lispobj` **&rest** *KEYS* **&key** *SHORT-NAME LONG-NAME DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR TYPESPEC FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

Make a new lispobj option.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash).

It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings.
It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable.
It defaults to nil.
- TYPESPEC is a type specifier the option's value should satisfy.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [`net.didierverna.clon`], page 27

Source [`lispobj`], page 16 (Lisp file)

`make-path` **&rest** *KEYS* **&key** *SHORT-NAME LONG-NAME DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR FALLBACK-VALUE DEFAULT-VALUE TYPE HIDDEN* [Function]

Make a new path option.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings.
It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable.
It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- TYPE is the pathname type. It can be one of :file, :directory, :file-list, :directory-list or nil meaning that everything is allowed.
- When HIDDEN, the option doesn't appear in help strings.

Package [`net.didierverna.clon`], page 27

Source [`path`], page 16 (Lisp file)

`make-stropt` **&rest** *KEYS* **&key** *SHORT-NAME LONG-NAME DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

Make a new string option.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings.
It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings.

- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable.
It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27

Source [stropt], page 16 (Lisp file)

`make-switch` **&rest** *KEYS* **&key** *SHORT-NAME LONG-NAME DESCRIPTION ARGUMENT-STYLE ARGUMENT-TYPE ENV-VAR DEFAULT-VALUE HIDDEN* [Function]

Make a new switch.
- SHORT-NAME is the switch's short name (without the dash).
It defaults to nil.
- LONG-NAME is the switch's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the switch's description appearing in help strings.
It defaults to nil.
- ARGUMENT-STYLE is the switch's argument display style. It can be one of :yes/no, :on/off, :true/false, :yup/nope or :yeah/nah.
It defaults to :yes/no.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the switch's associated environment variable.
It defaults to nil.
- DEFAULT-VALUE is the switch's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27

Source [switch], page 15 (Lisp file)

`make-synopsis` **&rest** *KEYS* **&key** *POSTFIX ITEM MAKE-DEFAULT* [Function]

Make a new SYNOPSIS.
- POSTFIX is a string to append to the program synopsis, in case it accepts a remainder.
- If MAKE-DEFAULT, make the new synopsis the default one.

Package [net.didierverna.clon], page 27

Source [synopsis], page 19 (Lisp file)

`make-text` **&rest** *KEYS* **&key** *CONTENTS HIDDEN* [Function]

Make a new text.
- CONTENTS is the actual text to display.
- When HIDDEN, the text doesn't appear in help strings.

Package [net.didierverna.clon], page 27

Source [text], page 13 (Lisp file)

`make-xswitch` **&rest** *KEYS* **&key** *SHORT-NAME LONG-NAME DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENUM ENV-VAR DEFAULT-VALUE HIDDEN* [Function]

Make a new xswitch.
- SHORT-NAME is the xswitch's short name (without the dash).
It defaults to nil.
- LONG-NAME is the xswitch's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the xswitch's description appearing in help strings.
It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENUM is the set of possible non-boolean values.
- ENV-VAR is the xswitch's associated environment variable.
It defaults to nil.
- DEFAULT-VALUE is the xswitch's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [`net.didierverna.clon`], page 27

Source [`xswitch`], page 17 (Lisp file)

`nickname-package` **&optional** *NICKNAME* [Function]

Add NICKNAME (:CLON by default) to the :NET.DIDIERVERNA.CLON package.

Package [`net.didierverna.clon`], page 27

Source [`meta`], page 11 (Lisp file)

`progname` **&key** *CONTEXT* [Function]

Return CONTEXT's program name.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`remainder` **&key** *CONTEXT* [Function]

Return CONTEXT's remainder.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`setup-termio` [Function]

Autodetect termio support.
Update Clon configuration and *FEATURES* accordingly.

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`version` **&optional** *TYPE* [Function]

Return the current version of Clon.
TYPE can be one of :number, :short or :long.

A version number is computed as major*10000 + minor*100 + patchlevel, leaving two digits for each level. Alpha, beta and rc status are ignored in version numbers.

A short version is something like 1.3{a,b,rc}4, or 1.3.4 for patchlevel. Alpha, beta or rc levels start at 1. Patchlevels start at 0 but are ignored
in the output, so that 1.3.0 appears as just 1.3.

A long version is something like
1.3 {alpha,beta,release candidate,patchlevel} 4 "Michael Brecker". As for the short version, a patchlevel of 0 is ignored in the output.

Package [net.didierverna.clon.setup], page 39

Source [setup], page 25 (Lisp file)

6.2 Internal definitions

6.2.1 Constants

`+tiocgwinsz+` [Constant]

NIL

Package [net.didierverna.clon], page 27

Source /Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier

`offset-of-winsize-ws-col` [Constant]

Package [net.didierverna.clon], page 27

Source /Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier

`offset-of-winsize-ws-row` [Constant]

Package [net.didierverna.clon], page 27

Source /Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier

`offset-of-winsize-ws-xpixel` [Constant]

Package [net.didierverna.clon], page 27

Source /Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier

`offset-of-winsize-ws-ypixel` [Constant]

Package [net.didierverna.clon], page 27

Source /Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier

`size-of-winsize` [Constant]

Package [net.didierverna.clon], page 27

Source /Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier

6.2.2 Special variables

`*executablep*` [Special Variable]

Whether the current Lisp image is a standalone executable dumped by Clon.
This information is needed in some implementations that treat their command-line differently in dumped images.

Package [net.didierverna.clon], page 27

Source [util], page 12 (Lisp file)

`*highlight-properties*` [Special Variable]
The highlight face properties.

Package [`net.didierverna.clon`], page 27

Source [`face`], page 20 (Lisp file)

`*item-names*` [Special Variable]
The list of defined item names.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

`*readtable*` [Special Variable]
The Clon readtable.

Package [`net.didierverna.clon`], page 27

Source [`meta`], page 11 (Lisp file)

6.2.3 Macros

`%defgroup` *INTERNALP* (**&rest** *KEYS* **&key** *HEADER HIDDEN*) **&body** *FORMS* [Macro]
Define a new group.

Package [`net.didierverna.clon`], page 27

Source [`group`], page 18 (Lisp file)

`accumulate` (*INITIAL-VALUE*) **&body** *BODY* **&aux** *PLACE INITIAL-PLACE* [Macro]
Accumulate BODY forms in a list beginning with INITIAL-VALUE.
INITIAL-VALUE is not evaluated. BODY forms are accumulated only when their value is non-nil.
If nothing to accumulate, then return nil instead of the list of INITIAL-VALUE.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`declare-valid-superclass` *CLASS SUPERCLASS* [Macro]
Validate SUPERCLASS classes for CLASS classes.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`defabstract` *CLASS SUPER-CLASSES SLOTS* **&rest** *OPTIONS* [Macro]
Like DEFCLASS, but define an abstract class.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`defindent` *SYMBOL INDENT* [Macro]
Set SYMBOL's indentation to INDENT in (X)Emacs.
SYMBOL and INDENT need not be quoted.
See CLINDENT for more information.

Package [`net.didierverna.clon`], page 27

Source [`meta`], page 11 (Lisp file)

`defoption` *CLASS SUPERCLASSES SLOTS* **&rest** *OPTIONS* [Macro]
Create a new option CLASS and register it with Clon.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

`do-options` (*OPT THERE*) **&body** *BODY* [Macro]
Execute BODY with OPT bound to every option in THERE.

Package [`net.didierverna.clon`], page 27

Source [`synopsis`], page 19 (Lisp file)

`econd` **&body** *CLAUSES* [Macro]
Like COND, but signal an error if no clause evaluates to t.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`endpush` *OBJECT PLACE* [Macro]
Like push, but at the end.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`highlight-property-ecase` *PROPERTY VALUE* **&body** *CLAUSES* [Macro]
Create an ECASE form to extract PROPERTY's VALUE escape sequence.
Each clause looks like: (PROPERTY-NAME (VALUE-OR-VALUE-LIST ESCAPE-SEQUENCE)*). The value-matching part will itself be enclosed in an ECASE expression.
In addition, the special clause syntax (BOOLEAN <PROPERTY-NAME> <YES> <NO>) is a shortcut for: (PROPERTY-NAME ((on t) YES) ((off nil) NO)).

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`in-readtable` *NAME* [Macro]
Set the current readtable to the value of NAME::*READTABLE*.

Package [`net.didierverna.clon`], page 27

Source [`meta`], page 11 (Lisp file)

`map-frames` *FUNCTION* (*SHEET* **&key** *REVERSE*) [Macro]
Map FUNCTION over SHEET's frames.
If REVERSE, map in reverse order.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`maybe-pop-argument` *CMDLINE OPTION CMDLINE-ARGUMENT* [Macro]
Pop OPTION's argument from CMDLINE if needed.
If so, store it in CMDLINE-ARGUMENT.

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

`maybe-push` *OBJECT PLACE* **&aux** *THE-OBJECT* [Macro]
Like push, but only if OBJECT is non-nil.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`replace-in-keys` (*KEY VAL*) *KEYS THE-KEY FORM* [Macro]
Replace every occurrence of THE-KEY in KEYS with FORM.
At every KEYS round, KEY and VAL are bound to the current key-value pair.
FORM is evaluated each time and should return a key-value list.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`restartable-invalid-negated-syntax-error` (*OPTION*) **&body** *BODY* [Macro]
Restartably throw an invalid-negated-syntax error.
The error relates to the command-line use of OPTION.
BODY constitutes the body of the only restart available,
use-short-call, and should act as if OPTION had been normally called by short name.

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

`restartable-spurious-cmdline-argument-error` (*OPTION NAME ARGUMENT*) **&body** *BODY* [Macro]
Restartably throw a spurious-cmdline-argument error.
The error relates to the command-line use of OPTION called by NAME with ARGUMENT.
BODY constitutes the body of the only restart available, discard-argument, and should act as if ARGUMENT had not been provided.

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

`with-context-error-handler` *CONTEXT* **&body** *BODY* [Macro]
Execute BODY with CONTEXT's error handler bound for CONDITION.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`with-winsize` *VAR35* (**&rest** *FIELD-VALUES-36*) **&body** *BODY37* [Macro]

Package [`net.didierverna.clon`], page 27

Source `/Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier`

6.2.4 Functions

`%version` *TYPE MAJOR MINOR STATUS LEVEL NAME* [Function]

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`add-subface` *FACE SUBFACE* [Function]
Add SUBFACE to FACE's subfaces and return it.

Package [`net.didierverna.clon`], page 27

Source [`face`], page 20 (Lisp file)

`allocate-winsize` [Function]

Package [net.didierverna.clon], page 27

Source /Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier

`argument-popable-p` *CMDLINE* [Function]

Return true if the first CMDLINE item is an argument.

Package [net.didierverna.clon], page 27

Source [cmdline], page 18 (Lisp file)

`attach-face-tree` *FACE FACE-TREE* **&aux** *NEW-TREE* [Function]

Create a copy of FACE-TREE, attach it to FACE and return it.
Apart from the parenting information, the copied faces share slot values with the original ones.

Package [net.didierverna.clon], page 27

Source [face], page 20 (Lisp file)

`available-right-margin` *SHEET* [Function]

Return SHEET's available right margin.
This margin is the first non-self margin specified by a frame. All inner self frames can potentially write until the available right margin.

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`beginning-of-string-p` *BEGINNING STRING* **&optional** *IGNORE-CASE* **&aux** *LENGTH* [Function]

Check that STRING starts with BEGINNING.
If IGNORE-CASE, well, ignore case.

Package [net.didierverna.clon], page 27

Source [util], page 12 (Lisp file)

`clindent` *SYMBOL INDENT* [Function]

Set SYMBOL's indentation to INDENT in (X)Emacs.
This function sets SYMBOL's common-lisp-indent-function property.
If INDENT is a symbol, use its indentation definition.
Otherwise, INDENT is considered as an indentation definition.

Package [net.didierverna.clon], page 27

Source [meta], page 11 (Lisp file)

`close-line` *SHEET* [Function]

Close all frames on SHEET's current line and go to next line.

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`close-sface` *SHEET* [Function]

Close SHEET's current sface.

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`closest-match` *MATCH LIST* **&key** *IGNORE-CASE KEY* **&aux** *MATCH-LENGTH SHORTEST-DISTANCE CLOSEST-MATCH* [Function]

Return the LIST element closest to MATCH, or nil.
If IGNORE-CASE, well, ignore case.
KEY should provide a way to get a string from each LIST element.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`cmdline-convert` *VALUED-OPTION CMDLINE-NAME CMDLINE-ARGUMENT* [Function]

Convert CMDLINE-ARGUMENT to VALUED-OPTION's value.
This function is used when the conversion comes from a command-line usage of VALUED-OPTION, called by CMDLINE-NAME, and intercepts invalid-argument errors to raise the higher level invalid-cmdline-argument error instead.

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

`cmdline-option-name` *INSTANCE* [Function]
`(setf cmdline-option-name)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`cmdline-option-option` *INSTANCE* [Function]
`(setf cmdline-option-option)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`cmdline-option-p` *OBJECT* [Function]

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`cmdline-option-source` *INSTANCE* [Function]
`(setf cmdline-option-source)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`cmdline-option-value` *INSTANCE* [Function]
`(setf cmdline-option-value)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`complete-string` *BEGINNING COMPLETE* [Function]

Complete BEGINNING with the rest of COMPLETE in parentheses.
For instance, completing 'he' with 'help' will produce 'he(lp)'.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`copy-cmdline-option` *INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`copy-frame` *INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`copy-highlight-frame` *INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`copy-highlight-property-instance` *INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`current-frame` *SHEET* [Function]

Return SHEET's current frame.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`current-left-margin` *SHEET* [Function]

Return SHEET's current left margin.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`current-right-margin` *SHEET* [Function]

Return SHEET's current right margin.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`current-sface` *SHEET* [Function]

Return SHEET's current sface or nil.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`directory-pathname-p` *PATHNAME* [Function]

Return true if PATHNAME denotes a directory.

Package [`net.didierverna.clon`], page 27

Source [`path`], page 16 (Lisp file)

`environment-convert` *VALUED-OPTION ENV-VAL* [Function]

Convert ENV-VAL to VALUED-OPTION's value.
This function is used when the conversion comes from an environment variable associated with VALUED-OPTION, and intercepts invalid-argument errors
to raise the higher level invalid-environment-value error instead.

Package [`net.didierverna.clon`], page 27

Source [`environ`], page 19 (Lisp file)

`exit-abnormally` *ERROR* [Function]
Print ERROR on *ERROR-OUTPUT* and exit with status code 1.

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

`face-highlight-property-set-p` *FACE PROPERTY* [Function]
Return t if PROPERTY is set explicitely in FACE.

Package [net.didierverna.clon], page 27

Source [face], page 20 (Lisp file)

`face-highlight-property-value` *FACE PROPERTY* [Function]
Return PROPERTY's value in FACE.
Since faces inherit highlight properties, the actual value might come from one of FACE's ancestors.
if PROPERTY is not et, return nil.

Package [net.didierverna.clon], page 27

Source [face], page 20 (Lisp file)

`find-sface` *SFACE NAME* **&aux** *SIBLING SUB-SFACE* [Function]
Find an sface starting at SFACE named NAME.
If the sface can't be found in SFACE's face tree, find one in SFACE's sibling instead, and make a copy of it.

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`flush-sheet` *SHEET* [Function]
Flush SHEET.

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`frame-left-margin` *INSTANCE* [Function]
`(setf frame-left-margin)` *VALUE INSTANCE* [Function]

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`frame-p` *OBJECT* [Function]

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`frame-right-margin` *INSTANCE* [Function]
`(setf frame-right-margin)` *VALUE INSTANCE* [Function]

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`frame-sface` *INSTANCE* [Function]
`(setf frame-sface)` *VALUE INSTANCE* [Function]

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`free-winsize` *OBJECT40* [Function]

Package [net.didierverna.clon], page 27

Source /Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier

`get-top-padding` *SFACE ITEMS* [Function]

Return top padding of the next item in ITEMS that will print under SFACE.

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`getenv` *VARIABLE* [Function]

Get environment VARIABLE's value. VARIABLE may be null.

Package [net.didierverna.clon], page 27

Source [util], page 12 (Lisp file)

`help-spec-items-will-print` *SFACE ITEMS* [Function]

Return t if at least one of ITEMS will print under SFACE.

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`highlight-frame-highlight-property-instances` *INSTANCE* [Function]
`(setf highlight-frame-highlight-property-instances)` *VALUE INSTANCE* [Function]

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`highlight-frame-left-margin` *INSTANCE* [Function]
`(setf highlight-frame-left-margin)` *VALUE INSTANCE* [Function]

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`highlight-frame-p` *OBJECT* [Function]

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`highlight-frame-right-margin` *INSTANCE* [Function]
`(setf highlight-frame-right-margin)` *VALUE INSTANCE* [Function]

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`highlight-frame-sface` *INSTANCE* [Function]
`(setf highlight-frame-sface)` *VALUE INSTANCE* [Function]

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`highlight-property-instance-escape-sequence` *INSTANCE* [Function]

Return highlight property INSTANCE's escape sequence.

Package [net.didierverna.clon], page 27

Source [sheet], page 21 (Lisp file)

`highlight-property-instance-name` *INSTANCE* [Function]
`(setf highlight-property-instance-name)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`highlight-property-instance-p` *OBJECT* [Function]

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`highlight-property-instance-value` *INSTANCE* [Function]
`(setf highlight-property-instance-value)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`home-directory` [Function]
Return user's home directory in canonical form.
If the user's home directory cannot be computed, signal a warning and return NIL.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`i-reader` *STREAM SUBCHAR ARG* [Function]
Read an argument list for the DEFINDENT macro.

Package [`net.didierverna.clon`], page 27

Source [`meta`], page 11 (Lisp file)

`list-to-string` *LIST* **&key** *KEY SEPARATOR* [Function]
Return a SEPARATOR-separated string of all LIST elements.
- KEY should provide a way to get a string from each LIST element.
- SEPARATOR is the string to insert between elements.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`macosp` [Function]
Return t if running on Mac OS.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`make-cmdline-option` **&key** (*NAME* **NAME**) (*OPTION* **OPTION**) [Function]
(*VALUE* **VALUE**) (*SOURCE* **SOURCE**)

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`make-frame` **&key** (*SFACE* **SFACE**) (*LEFT-MARGIN* **LEFT-MARGIN**) [Function]
(*RIGHT-MARGIN* **RIGHT-MARGIN**)

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`make-highlight-frame` **&key** (*SFACE* **SFACE**) (*LEFT-MARGIN* **LEFT-MARGIN**) (*RIGHT-MARGIN* **RIGHT-MARGIN**) (*HIGHLIGHT-PROPERTY-INSTANCES* **HIGHLIGHT-PROPERTY-INSTANCES**) [Function]

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`make-highlight-property-instance` **&key** (*NAME* **NAME**) (*VALUE* **VALUE**) [Function]

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`make-internal-enum` *LONG-NAME DESCRIPTION* **&rest** *KEYS* **&key** *ARGUMENT-NAME ARGUMENT-TYPE ENUM ENV-VAR FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

Make a new internal (Clon-specific) enum option.
- LONG-NAME is the option's long-name, sans the 'clon-' prefix.
(Internal options don't have short names.)
- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENUM is the set of possible values.
- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [`net.didierverna.clon`], page 27

Source [`enum`], page 17 (Lisp file)

`make-internal-flag` *LONG-NAME DESCRIPTION* **&rest** *KEYS* **&key** *ENV-VAR HIDDEN* [Function]

Make a new internal (Clon-specific) flag.
- LONG-NAME is the flag's long-name, sans the 'clon-' prefix.
(Internal options don't have short names.)
- DESCRIPTION is the flag's description.
- ENV-VAR is the flag's associated environment variable, sans the 'CLON_' prefix. It default to nil.
- When HIDDEN, the option doesn't appear in help strings.

Package [`net.didierverna.clon`], page 27

Source [`flag`], page 14 (Lisp file)

`make-internal-lispobj` *LONG-NAME DESCRIPTION* **&rest** *KEYS* **&key** *ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR TYPESPEC FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

Make a new internal (Clon-specific) string option.
- LONG-NAME is the option's long-name, sans the 'clon-' prefix.
(Internal options don't have short names.)

- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- TYPESPEC is a type specifier the option's value should satisfy.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [`net.didierverna.clon`], page 27

Source [`lispobj`], page 16 (Lisp file)

`make-internal-path` *LONG-NAME DESCRIPTION* **&rest** *KEYS* **&key** *ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR FALLBACK-VALUE DEFAULT-VALUE TYPE HIDDEN* [Function]

Make a new internal (Clon-specific) path option.
- LONG-NAME is the option's long-name, sans the 'clon-' prefix.
(Internal options don't have short names.)
- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- TYPE is the pathname type. It can be one of :file, :directory, :file-list, :directory-list or nil meaning that everything is allowed.
- When HIDDEN, the option doesn't appear in help strings.

Package [`net.didierverna.clon`], page 27

Source [`path`], page 16 (Lisp file)

`make-internal-stropt` *LONG-NAME DESCRIPTION* **&rest** *KEYS* **&key** *ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

Make a new internal (Clon-specific) string option.
- LONG-NAME is the option's long-name, sans the 'clon-' prefix.
(Internal options don't have short names.)
- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27

Source [stropt], page 16 (Lisp file)

`make-internal-switch` *LONG-NAME DESCRIPTION* **&rest** *KEYS* **&key** *ARGUMENT-STYLE ARGUMENT-TYPE ENV-VAR DEFAULT-VALUE HIDDEN* [Function]

Make a new internal (Clon-specific) switch.
- LONG-NAME is the switch's long-name, sans the 'clon-' prefix.
(Internal options don't have short names.)
- DESCRIPTION is the switch's description.
- ARGUMENT-STYLE is the switch's argument display style. It can be one of :yes/no, :on/off, :true/false, :yup/nope or :yeah/nah.
It defaults to :yes/no.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the switch's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- DEFAULT-VALUE is the switch's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27

Source [switch], page 15 (Lisp file)

`make-internal-text` **&rest** *KEYS* **&key** *CONTENTS HIDDEN* [Function]

Package [net.didierverna.clon], page 27

Source [text], page 13 (Lisp file)

`make-internal-xswitch` *LONG-NAME DESCRIPTION* **&rest** *KEYS* **&key** *ARGUMENT-NAME ARGUMENT-TYPE ENUM ENV-VAR DEFAULT-VALUE HIDDEN* [Function]

Make a new internal (Clon-specific) xswitch.
- LONG-NAME is the xswitch's long-name, sans the 'clon-' prefix.
(Internal options don't have short names.)
- DESCRIPTION is the xswitch's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENUM is the set of possible non-boolean values.
- ENV-VAR is the xswitch's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- DEFAULT-VALUE is the xswitch's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27

Source [xswitch], page 17 (Lisp file)

`make-raw-face-tree` **&optional** *FACE-CLASS* [Function]

Make a raw (boring yet functional) face tree.

Package [net.didierverna.clon], page 27

Source [face], page 20 (Lisp file)

`make-raw-sface` *SIBLING* **&aux** *SFACE* [Function]
Return a new SFace based on SIBLING.
This function does not consider SIBLING as a face tree:
only face properties are copied; the face parent and children are set to nil.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`make-sheet` **&rest** *KEYS* **&key** *OUTPUT-STREAM SEARCH-PATH THEME LINE-WIDTH HIGHLIGHT* [Function]
Make a new SHEET.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`match-option` *OPTION* **&key** *SHORT-NAME LONG-NAME* [Function]
Try to match OPTION against SHORT-NAME, LONG-NAME.
If OPTION matches, return the name that matched.

Package [`net.didierverna.clon`], page 27

Source [`option`], page 13 (Lisp file)

`open-line` *SHEET* [Function]
Open all frames on SHEET's current line.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`open-next-line` *SHEET* [Function]
Close SHEET's current line and open the next one.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`open-sface` *SHEET SFACE* [Function]
Create a frame for SFACE and open it.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`option-abbreviation-distance` *OPTION PARTIAL-NAME* [Function]
Return the distance between OPTION's long name and PARTIAL-NAME.
If PARTIAL-NAME does not abbreviate OPTION's long name, return MOST-POSITIVE-FIXNUM.

Package [`net.didierverna.clon`], page 27

Source [`option`], page 13 (Lisp file)

`option-call-p` *STR* [Function]
Return true if STR looks like an option call.

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

`parent-generation` *FACE PARENT-NAME* [Function]
Return FACE's parent generation for PARENT-NAME.
That is, 1 if PARENT-NAME names FACE's parent, 2 if it names its grand-parent etc. If PARENT-NAME does not name one of FACE's ancestors, trigger an error.

Package [`net.didierverna.clon`], page 27

Source [`face`], page 20 (Lisp file)

`pathname-component-null-p` *COMPONENT* [Function]
Return true if COMPONENT is either null or :unspecific.

Package [`net.didierverna.clon`], page 27

Source [`path`], page 16 (Lisp file)

`pop-frame` *SHEET* [Function]
Pop SHEET's current frame.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`potential-pack-char` *OPTION* **&optional** *AS-STRING* [Function]
Return OPTION's potential pack character, if any.
If AS-STRING, return a string of that character.

Package [`net.didierverna.clon`], page 27

Source [`option`], page 13 (Lisp file)

`princ-char` *SHEET CHAR* [Function]
Princ CHAR on SHEET's stream and increment the column position.
The effect of printing CHAR must be exactly to move right by one column, so control characters, as well as newlines and tabs are forbidden here.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`princ-highlight-property-instances` *SHEET INSTANCES* [Function]
Princ highlight proeprty INSTANCES on SHEET's stream.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`princ-spaces` *SHEET NUMBER* [Function]
Princ NUMBER spaces to SHEET's stream and update the column position.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`princ-string` *SHEET STRING* [Function]
Princ STRING on SHEET's stream and update the column position.
The effect of printing STRING must be exactly to move right by the corresponding string length, so control characters, as well as newlines and tabs are forbidden here.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`print-error` *ERROR* **&optional** *INTERACTIVEP* **&aux** *STREAM* **PRINT-ESCAPE** [Function]
Print ERROR on *ERROR-OUTPUT*.
When INTERACTIVEP, print on *QUERY-IO* instead.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`print-faced-help-spec` *SHEET SFACE ITEMS* [Function]
Print all help specification ITEMS on SHEET with SFACE.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`print-help` *SHEET HELP* [Function]
Open the toplevel help face and print HELP on SHEET with it.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`print-string` *SHEET STRING* [Function]
Output STRING to SHEET.
STRING is output within the current frame's bounds.
Spacing characters are honored but newlines might replace spaces when the output reaches the rightmost bound.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`push-frame` *SHEET FRAME* [Function]
Push a new frame to SHEET's frames.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`putenv` *VARIABLE VALUE* [Function]
Set environment VARIABLE to VALUE.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`reach-column` *SHEET COLUMN* [Function]
Reach COLUMN on SHEET by princ'ing spaces.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`read-argument` [Function]
Read an option argument from standard input.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

`read-call` **&optional** *NEGATED* [Function]
Read an option's call or pack from standard input.
If NEGATED, read a negated call or pack. Otherwise, read a short call or pack.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`read-env-val` *ENV-VAR* [Function]
Read ENV-VAR's new value from standard input.

Package [`net.didierverna.clon`], page 27

Source [`environ`], page 19 (Lisp file)

`read-long-name` [Function]
Read an option's long name from standard input.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`read-sface-tree` *PATHNAME* [Function]
Read an sface tree from PATHNAME.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`read-value` [Function]
Read an option value from standard input.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

`release-status-number` *RELEASE-STATUS* [Function]

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`remove-keys` *KEYS* **&rest** *REMOVED* [Function]
Return a new property list from KEYS without REMOVED ones.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`replace-key` *REPLACEMENT KEYS* [Function]
Return a new property list from KEYS with REPLACEMENT.
REPLACEMENT can take the following forms:
- :KEY
The effect is to remove :KEY from KEYS, as per REMOVE-KEYS.
- (:KEY :NEW-KEY)
The effect is to replace :KEY with :NEW-KEY, leaving the values unchanged.
- (:KEY :NEW-KEY (VAL-OR-VALS NEW-VAL)*), with VAL-OR-VALS being
either a value or a list of values. The effect is to replace :KEY with :NEW-KEY and a value matching one of the VAL-OR-VALS with the
corresponding NEW-VAL. Values not matching any VAL-OR-VALS remain unchanged. - (:KEY (VAL-OR-VALS :NEW-KEY NEW-VAL...)*), with VAL-OR-VALS as above. The effect is the same as above, but :NEW-KEY additionally depends on the matched value. If

multiple :NEW-KEY NEW-VAL couples are provided, that many new keys are inserted along with their values. For values not matching any VAL-OR-VALS, :KEY and its value remain unchanged.

Package [net.didierverna.clon], page 27

Source [util], page 12 (Lisp file)

`replace-keys` *KEYS* **&rest** *REPLACEMENTS* **&aux** *NEW-KEYS* [Function]

Return a new property list from KEYS with REPLACEMENTS.
See REPLACE-KEY for more information on the replacement syntax.

Package [net.didierverna.clon], page 27

Source [util], page 12 (Lisp file)

`restart-on-error` *ERROR* [Function]

Print ERROR and offer available restarts on *QUERY-IO*.

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

`restartable-check` *VALUED-OPTION VALUE* [Function]

Restartably check that VALUE is valid for VALUED-OPTION.
The only restart available, use-value, offers to try a different value from the one that was provided.

Package [net.didierverna.clon], page 27

Source [valued], page 14 (Lisp file)

`restartable-cmdline-convert` *VALUED-OPTION CMDLINE-NAME CMDLINE-ARGUMENT* [Function]

Restartably convert CMDLINE-ARGUMENT to VALUED-OPTION's value.
This function is used when the conversion comes from a command-line usage of VALUED-OPTION, called by CMDLINE-NAME.

As well as conversion errors, this function might raise a missing-cmdline-argument error if CMDLINE-ARGUMENT is nil and an argument is required.

Available restarts are (depending on the context):
- use-fallback-value: return FALLBACK-VALUE,
- use-default-value: return VALUED-OPTION's default value,
- use-value: return another (already converted) value,
- use-argument: return the conversion of another argument.

Return two values: VALUED-OPTION's value and the actual value source.
The value source may be :cmdline, :fallback or :default.

Package [net.didierverna.clon], page 27

Source [cmdline], page 18 (Lisp file)

`restartable-cmdline-junk-error` *JUNK* [Function]

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

`restartable-convert` *VALUED-OPTION ARGUMENT* [Function]
Restartably convert ARGUMENT to VALUED-OPTION's value.
Available restarts are:
- use-default-value: return OPTION's default value,
- use-value: return another (already converted) value,
- use-argument: return the conversion of another argument.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

`restartable-environment-convert` *VALUED-OPTION ENV-VAL* [Function]
Restartably convert ENV-VAL to VALUED-OPTION's value.
This function is used when the conversion comes from an environment variable associated with VALUED-OPTION.

Available restarts are:
- use-default-value: return VALUED-OPTION's default value,
- use-value: return another (already converted) value,
- use-argument: return the conversion of another argument,
- modify-env: modify the environment variable's value.

Package [`net.didierverna.clon`], page 27

Source [`environ`], page 19 (Lisp file)

`restrict-because` *REASON* [Function]
Put Clon in restricted mode because of REASON.

Package [`net.didierverna.clon.setup`], page 39

Source [`setup`], page 25 (Lisp file)

`safe-left-margin` *SHEET MARGIN* [Function]
Return either MARGIN or a safe value instead.
To be safe, margin must be greater than the current left margin and smaller than the currently available margin.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`safe-right-margin` *SHEET LEFT-MARGIN MARGIN* [Function]
Return either MARGIN or a safe value instead.
To be safe, margin must be greater than LEFT-MARGIN and smaller than the currently available right margin.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`search-branch` *FACE NAMES* [Function]
Search for a branch of faces named NAMES starting at FACE.
The branch is searched for as a direct subbranch of FACE, or as a direct subbranch of FACE's ancestors.
If a branch is found, return its leaf face. Otherwise return nil.

Package [`net.didierverna.clon`], page 27

Source [`face`], page 20 (Lisp file)

`search-face` *FACE NAME* **&optional** *ERROR-ME* [Function]
Search for a face named NAME starting at FACE.
The face is looked for as a direct subface of FACE (in which case it is simply returned), or up in the hierarchy and by successive upper branches (in which case it is copied and attached to FACE).
If ERROR-ME, trigger an error if no face is found; otherwise, return nil.

Package [`net.didierverna.clon`], page 27

Source [`face`], page 20 (Lisp file)

`search-option` *CONTEXT* **&rest** *KEYS* **&key** *SHORT-NAME* *LONG-NAME PARTIAL-NAME* [Function]
Search for an option in CONTEXT.
The search is done with SHORT-NAME, LONG-NAME, or PARTIAL-NAME.
In case of a PARTIAL-NAME search, look for an option the long name of which begins with it.
In case of multiple matches by PARTIAL-NAME, the longest match is selected. When such an option exists, return wo values:
- the option itself,
- the name used to find the option, possibly completed if partial.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`search-option-by-abbreviation` *CONTEXT PARTIAL-NAME* [Function]
Search for option abbreviated with PARTIAL-NAME in CONTEXT.
When such an option exists, return two values:
- the option itself,
- the completed name.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`search-option-by-name` *CONTEXT* **&rest** *KEYS* **&key** *SHORT-NAME* *LONG-NAME* [Function]
Search for option with either SHORT-NAME or LONG-NAME in CONTEXT.
When such an option exists, return two values:
- the option itself,
- the name that matched.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`search-sticky-option` *CONTEXT NAMEARG* [Function]
Search for a sticky option in CONTEXT, matching NAMEARG.
NAMEARG is the concatenation of the option's short name and its argument.
In case of multiple matches, the option with the longest name is selected. When such an option exists, return two values:
- the option itself,
- the argument part of NAMEARG.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

`select-keys` *KEYS* **&rest** *SELECTED* [Function]
Return a new property list from KEYS with only SELECTED ones.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

`split-path` *PATH* [Function]
Split PATH into a list of directories.

Package [`net.didierverna.clon`], page 27

Source [`path`], page 16 (Lisp file)

`stream-line-width` *STREAM* **&aux** *HANDLE* [Function]
Get STREAM's line width.
Return two values:
- the stream's line width, or nil if it can't be computed
(typically when the stream does not denote a tty),
- an error message if the operation failed.

Package [`net.didierverna.clon`], page 27

Source [`termio`], page 11 (Lisp file)

`tilde-reader` *STREAM CHAR* [Function]
Read a series of ~"string" to be concatenated together.

Package [`net.didierverna.clon`], page 27

Source [`meta`], page 11 (Lisp file)

`try-read-sface-tree` *PATHNAME* [Function]
Read an sface tree from PATHNAME if it exists or return nil.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`try-read-theme` *PATHNAME* [Function]
Read a theme from PATHNAME or PATHNAME.cth if it exists or return nil.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

`winsize-ws-col` *STRUCT* [Function]
`(setf winsize-ws-col)` *VAR-32 STRUCT* [Function]

Package [`net.didierverna.clon`], page 27

Source `/Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier`

`winsize-ws-row` *STRUCT* [Function]
`(setf winsize-ws-row)` *VAR-0 STRUCT* [Function]

Package [`net.didierverna.clon`], page 27

Source `/Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier`

`winsize-ws-xpixel` *STRUCT* [Function]
`(setf winsize-ws-xpixel)` *VAR-33 STRUCT* [Function]

Package [`net.didierverna.clon`], page 27

Source `/Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier`

`winsize-ws-ypixel` *STRUCT* [Function]
`(setf winsize-ws-ypixel)` *VAR-34 STRUCT* [Function]

Package [`net.didierverna.clon`], page 27

Source `/Users/didier/.cache/common-lisp/sbcl-1.2.13.35-6102b24-macosx-x64/Users/didier`

6.2.5 Generic functions

`argument` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`argument` (*CONDITION* `unknown-cmdline-option-error`) [Method]

`argument` (*CONDITION* `spurious-cmdline-argument`) [Method]

`argument` (*CONDITION* `invalid-argument`) [Method]

`argument-name` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`argument-name` (*VALUED-OPTION* `valued-option`) [Method]
The option's argument display name.

Source [`valued`], page 14 (Lisp file)

`argument-required-p` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`argument-required-p` (*VALUED-OPTION* `valued-option`) [Method]
Whether the option's argument is required.

Source [`valued`], page 14 (Lisp file)

`argument-style` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`argument-style` (*SWITCH-BASE* `switch-base`) [Method]
The selected argument style.

Source [`switch-base`], page 15 (Lisp file)

`argument-styles` *OBJECT* [Generic Function]
`(setf argument-styles)` *NEW-VALUE OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`argument-styles` (*SWITCH-BASE* `switch-base`) [Method]
`(setf argument-styles)` *NEW-VALUE* (*SWITCH-BASE* `switch-base`) [Method]
The possible argument styles.
The position of every argument style in the list must correspond to the position of the associated strings in the yes-values and no-values slots.

Source [`switch-base`], page 15 (Lisp file)

`background` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`background` (*FACE* `face`) [Method]
The face background.

Source [`face`], page 20 (Lisp file)

`blink` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`blink` (*FACE* `face`) [Method]
The face's blink speed.

Source [`face`], page 20 (Lisp file)

`bottom-padding` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`bottom-padding` (*FACE* `face`) [Method]
The face bottom padding.
This property can take the following forms:
- nil: the next output can start right at the end of this face's,
- 0: the next output should start on the next line,
- N>0: there should be N empty lines before the next output.

Source [`face`], page 20 (Lisp file)

`check` *VALUED-OPTION VALUE* [Generic Function]
Check that VALUE is valid for VALUED-OPTION.
If VALUE is valid, return it. Otherwise, raise an invalid-value error.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

Methods

`check` (*XSWITCH* `xswitch`) *VALUE* [Method]
Check that VALUE is valid for XSWITCH.

Source [`xswitch`], page 17 (Lisp file)

`check` (*ENUM* `enum`) *VALUE* [Method]
Check that VALUE is valid for ENUM.

Source [`enum`], page 17 (Lisp file)

`check` (*PATH* `path`) *VALUE* [Method]
Check that VALUE is valid for PATH.

Source [`path`], page 16 (Lisp file)

`check` (*LISPOBJ* `lispobj`) *VALUE* [Method]
Check that VALUE is valid for LISPOBJ.

Source [`lispobj`], page 16 (Lisp file)

`check` (*STROPT* `stropt`) *VALUE* [Method]
Check that VALUE is valid for STROPT.

Source [`stropt`], page 16 (Lisp file)

`check` (*SWITCH* `switch`) *VALUE* [Method]
Check that VALUE is valid for SWITCH.

Source [`switch`], page 15 (Lisp file)

`check-name-clash` *ITEM1 ITEM2* [Generic Function]
Check for name clash between ITEM1's options and ITEM2's options.

Package [`net.didierverna.clon`], page 27

Source [`option`], page 13 (Lisp file)

Methods

`check-name-clash` (*CONTAINER1* `container`) (*CONTAINER2* `container`) [Method]
Check for name clash between CONTAINER1's options and CONTAINER2's ones.

Source [`container`], page 18 (Lisp file)

`check-name-clash` *ITEM1* (*CONTAINER* `container`) [Method]
Check for name clash between ITEM1's options and CONTAINER's ones.

Source [`container`], page 18 (Lisp file)

`check-name-clash` (*CONTAINER* `container`) *ITEM2* [Method]
Check for name clash between CONTAINER's options and ITEM2's ones.

Source [`container`], page 18 (Lisp file)

`check-name-clash` *ITEM1* (*TEXT* `text`) [Method]
Do nothing (no name clash with a text object.

`check-name-clash` (*TEXT* `text`) *ITEM2* [Method]
Do nothing (no name clash with a text object.

`check-name-clash` (*OPTION1* `option`) (*OPTION2* `option`) [Method]
Ensure that there is no name clash between OPTION1 and OPTION2.

`clon-options-group` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`clon-options-group` (*CONTEXT* `context`) [Method]
Return the Clon options group of CONTEXT's synopsis.

Source [`context`], page 23 (Lisp file)

`clon-options-group` (*SYNOPSIS* `synopsis`) [Method]
The Clon options group.

Source [`synopsis`], page 19 (Lisp file)

`close-frame` *SHEET FRAME* [Generic Function]
Close FRAME on SHEET.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Method Combination
`progn`
Options: `:most-specific-last`

Methods

`close-frame` *SHEET* (*FRAME* `frame`) **&aux** *RIGHT-MARGIN* `progn` [Method]
Reach FRAME's right margin if it has one.

`close-frame` *SHEET* (*FRAME* `highlight-frame`) `progn` [Method]
Restore the upper frame's highlight properties.

`cmdline-options` *OBJECT* [Generic Function]
`(setf cmdline-options)` *NEW-VALUE OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`cmdline-options` (*CONTEXT* `context`) [Method]
`(setf cmdline-options)` *NEW-VALUE* (*CONTEXT* `context`) [Method]
The options from the command-line.

Source [`context`], page 23 (Lisp file)

`column` *OBJECT* [Generic Function]
`(setf column)` *NEW-VALUE OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`column` (*SHEET* `sheet`) [Method]
`(setf column)` *NEW-VALUE* (*SHEET* `sheet`) [Method]
The sheet's current column.

Source [`sheet`], page 21 (Lisp file)

`comment` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`comment` (*CONDITION* `invalid-argument`) [Method]

`comment` (*CONDITION* `invalid-value`) [Method]

`concealedp` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`concealedp` (*FACE* `face`) [Method]
The face's concealed status.

Source [`face`], page 20 (Lisp file)

`contents` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`contents` (*TEXT* `text`) [Method]
The actual text string.

Source [`text`], page 13 (Lisp file)

`convert` *VALUED-OPTION ARGUMENT* [Generic Function]
Convert ARGUMENT to VALUED-OPTION's value.
If ARGUMENT is invalid, raise an invalid-argument error.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

Methods

`convert` (*XSWITCH* `xswitch`) *ARGUMENT* [Method]
Convert ARGUMENT to an XSWITCH value.

Source [`xswitch`], page 17 (Lisp file)

`convert` (*ENUM* `enum`) *ARGUMENT* [Method]
Convert ARGUMENT to an ENUM value.

Source [`enum`], page 17 (Lisp file)

`convert` (*PATH* `path`) *ARGUMENT* [Method]
Convert ARGUMENT to a PATH value.

Source [`path`], page 16 (Lisp file)

`convert` (*LISPOBJ* `lispobj`) *ARGUMENT* [Method]
Convert ARGUMENT to a LISPOBJ value.

Source [`lispobj`], page 16 (Lisp file)

`convert` (*STROPT* `stropt`) *ARGUMENT* [Method]
Convert ARGUMENT to an STROPT value.

Source [`stropt`], page 16 (Lisp file)

`convert` (*SWITCH* `switch`) *ARGUMENT* [Method]
Convert ARGUMENT to a SWITCH value.

Source [`switch`], page 15 (Lisp file)

`copy-instance` *INSTANCE* **&optional** *SUBCLASS* [Generic Function]
Return a copy of INSTANCE.
Copy is either an object of INSTANCE's class, or INSTANCE's SUBCLASS if given.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

Methods

`copy-instance` *INSTANCE* **&optional** *SUBCLASS* [Method]
Return a copy of INSTANCE.
Both instances share the same slot values.

`crossed-out-p` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`crossed-out-p` (*FACE* `face`) [Method]
The face's crossed out status.

Source [`face`], page 20 (Lisp file)

`default-value` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`default-value` (*VALUED-OPTION* `valued-option`) [Method]
The option's default value.

Source [`valued`], page 14 (Lisp file)

`description` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`description` (*OPTION* `option`) [Method]
The option's description.

Source [`option`], page 13 (Lisp file)

`enum` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`enum` (*ENUM-BASE* `enum-base`) [Method]
The set of possible values.

Source [`enum-base`], page 17 (Lisp file)

`env-val` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`env-val` (*CONDITION* `invalid-environment-value`) [Method]

`env-var` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`env-var` (*CONDITION* `environment-error`) [Method]

`env-var` (*OPTION* `option`) [Method]
The option's associated environment variable.

Source [`option`], page 13 (Lisp file)

`error-handler` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`error-handler` (*CONTEXT* `context`) [Method]
The behavior to adopt on option retrieval errors.

Source [`context`], page 23 (Lisp file)

`error-string` *CONDITION* [Generic Function]
`(setf error-string)` *NEW-VALUE CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`error-string` (*CONDITION* `home-directory`) [Method]
`(setf error-string)` *NEW-VALUE* (*CONDITION* `home-directory`) [Method]

`fallback-value` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`fallback-value` (*VALUED-OPTION* `valued-option`) [Method]
The option's fallback value.

Source [`valued`], page 14 (Lisp file)

`foreground` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`foreground` (*FACE* `face`) [Method]
The face foreground.

Source [`face`], page 20 (Lisp file)

`framedp` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`framedp` (*FACE* `face`) [Method]
The face's framed status.

Source [`face`], page 20 (Lisp file)

`frames` *OBJECT* [Generic Function]
`(setf frames)` *NEW-VALUE OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`frames` (*SHEET* `sheet`) [Method]
`(setf frames)` *NEW-VALUE* (*SHEET* `sheet`) [Method]
The stack of currently open frames.

Source [`sheet`], page 21 (Lisp file)

`get-bottom-padding` *SFACE HELP-SPEC* [Generic Function]
Get HELP-SPEC's bottom-padding under SFACE.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Methods

`get-bottom-padding` *SFACE HELP-SPEC* [Method]
Basic help specifications (chars, strings etc) don't provide a bottom padding.

`get-bottom-padding` *SFACE* (*HELP-SPEC* `list`) [Method]
Return the bottom padding of HELP-SPEC's face.

`header` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`header` (*GROUP* `group`) [Method]
The group's header.

Source [`group`], page 18 (Lisp file)

`help-spec` *ITEM* **&key** *UNHIDE PROGRAM* **&allow-other-keys** [Generic Function]
Return ITEM's help specification.

Package [`net.didierverna.clon`], page 27

Source [`item`], page 12 (Lisp file)

Methods

`help-spec` (*SYNOPSIS* `synopsis`) **&key** *PROGRAM* [Method]
Return SYNOPSIS's help specification.

Source [`synopsis`], page 19 (Lisp file)

`help-spec` (*GROUP* `group`) **&key** [Method]
Return GROUP's help specification.

Source [`group`], page 18 (Lisp file)

`help-spec` (*CONTAINER* `container`) **&key** [Method]
Return CONTAINER's help specification.

Source [`container`], page 18 (Lisp file)

`help-spec` (*OPTION* `valued-option`) **&key** [Method]
Return OPTION's help specification.

Source [`valued`], page 14 (Lisp file)

`help-spec` (*OPTION* `option`) **&key** [Method]
Return OPTION's help specification.

Source [`option`], page 13 (Lisp file)

`help-spec` (*TEXT* `text`) **&key** [Method]
Return TEXT's help specification.

Source [`text`], page 13 (Lisp file)

`help-spec` (*ITEM* `item`) **&key** *UNHIDE* `around` [Method]
Call the actual method only when ITEM is not hidden or UNHIDE.

`help-spec-will-print` *SFACE HELP-SPEC* [Generic Function]
Return t if HELP-SPEC will print under FACE.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Methods

`help-spec-will-print` *SFACE HELP-SPEC* `before` [Method]

`help-spec-will-print` *SFACE HELP-SPEC* [Method]
Basic help specifications (chars, strings etc) do print.

`help-spec-will-print` *SFACE* (*HELP-SPEC* `list`) [Method]
Return t if HELP-SPEC's items will print under HELP-SPEC's face.

`hiddenp` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`hiddenp` (*ITEM* `item`) [Method]
Whether the item is hidden in help strings.

Source [`item`], page 12 (Lisp file)

`highlight` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`highlight` (*CONTEXT* `context`) [Method]
Clon's output highlight mode.

Source [`context`], page 23 (Lisp file)

`highlightp` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`highlightp` (*SHEET* `sheet`) [Method]
Whether to highlight SHEET's output.

Source [`sheet`], page 21 (Lisp file)

`intensity` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`intensity` (*FACE* `face`) [Method]
The face intensity.

Source [`face`], page 20 (Lisp file)

`inversep` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`inversep` (*FACE* `face`) [Method]
The face's inverse video status.

Source [`face`], page 20 (Lisp file)

`italicp` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`italicp` (*FACE* `face`) [Method]
The face's italic status.

Source [`face`], page 20 (Lisp file)

`item` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`item` (*CONDITION* `cmdline-error`) [Method]

`item-separator` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`item-separator` (*FACE* `face`) [Method]
The face item separator.

Source [`face`], page 20 (Lisp file)

`items` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`items` (*CONTAINER* `container`) [Method]
The items in the container.

Source [`container`], page 18 (Lisp file)

`junk` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`junk` (*CONDITION* `cmdline-junk-error`) [Method]

`left-padding` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`left-padding` (*FACE* `face`) [Method]
The face left padding.
This property can take the following forms:
- <NUMBER>: the padding is relative to the enclosing face,
- SELF: the padding is set to wherever the face happens to be opened,
- (<NUMBER> ABSOLUTE): the padding is set in absolute value,
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.

Source [face], page 20 (Lisp file)

`line-width` *OBJECT* [Generic Function]

Package [net.didierverna.clon], page 27

Methods

`line-width` (*CONTEXT* `context`) [Method]
The line width for help display.

Source [context], page 23 (Lisp file)

`line-width` (*SHEET* `sheet`) [Method]
The sheet's line width.

Source [sheet], page 21 (Lisp file)

`long-name` *OBJECT* [Generic Function]

Package [net.didierverna.clon], page 27

Methods

`long-name` (*OPTION* `option`) [Method]
The option's long name.

Source [option], page 13 (Lisp file)

`make-face-tree` *DEFINITION* **&optional** *FACE-CLASS* [Generic Function]
Make a FACE-CLASS face tree from DEFINITION.

Package [net.didierverna.clon], page 27

Source [face], page 20 (Lisp file)

Methods

`make-face-tree` (*DEFINITION* `list`) **&optional** *FACE-CLASS* [Method]
Make a FACE-CLASS face tree from a list of face name and initargs.

`make-face-tree` (*NAME* `symbol`) **&optional** *FACE-CLASS* [Method]
Create a face named NAME.

`mapoptions` *FUNC THERE* [Generic Function]
Map FUNC over all options in THERE.

Package [net.didierverna.clon], page 27

Source [synopsis], page 19 (Lisp file)

Methods

`mapoptions` *FUNC* (*CONTEXT* `context`) [Method]
Map FUNC over all options in CONTEXT synopsis.

Source [`context`], page 23 (Lisp file)

`mapoptions` *FUNC ELSEWHERE* [Method]
Do nothing by default.

`mapoptions` *FUNC* (*ITEM* `item`) `after` [Method]
Mark TRAVERSABLE as traversed.

`mapoptions` *FUNC* (*CONTAINER* `container`) [Method]
Map FUNC over all containers or options in CONTAINER.

`mapoptions` *FUNC* (*OPTION* `option`) [Method]
Call FUNC on OPTION.

`name` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`name` (*CONDITION* `unknown-cmdline-option-error`) [Method]

`name` (*FACE* `face`) [Method]
The face name.

Source [`face`], page 20 (Lisp file)

`name` (*CONDITION* `cmdline-option-error`) [Method]

`negated-call` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`negated-call` (*CONDITION* `unrecognized-negated-call-error`) [Method]

`negated-pack` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`negated-pack` (*CONTEXT* `context`) [Method]
Return the negated pack of CONTEXT's synopsis.

Source [`context`], page 23 (Lisp file)

`negated-pack` (*SYNOPSIS* `synopsis`) [Method]
The negated pack string.

Source [`synopsis`], page 19 (Lisp file)

`negated-pack-char` *OPTION* **&optional** *AS-STRING* [Generic Function]
Return OPTION's negated pack character, if any.
If AS-STRING, return a string of that character.

Package [`net.didierverna.clon`], page 27

Source [`option`], page 13 (Lisp file)

Methods

`negated-pack-char` (*NEGATABLE* `negatable`) **&optional** *AS-STRING* [Method]
Return NEGATABLE's negated pack character, if any.

Source [`negatable`], page 15 (Lisp file)

`negated-pack-char` (*OPTION* `option`) **&optional** *AS-STRING* [Method]
Return nil (only the switch hierarchy is negated-pack'able).

`no-values` *OBJECT* [Generic Function]
`(setf no-values)` *NEW-VALUE OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`no-values` (*SWITCH-BASE* `switch-base`) [Method]
`(setf no-values)` *NEW-VALUE* (*SWITCH-BASE* `switch-base`) [Method]
The possible 'no' values.

Source [`switch-base`], page 15 (Lisp file)

`open-frame` *SHEET FRAME* [Generic Function]
Open FRAME on SHEET.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Method Combination
`progn`
Options: `:most-specific-last`

Methods

`open-frame` *SHEET* (*FRAME* `frame`) `progn` [Method]
Reach the frame's left margin.

`open-frame` *SHEET* (*FRAME* `highlight-frame`) `progn` [Method]
Reach the frame's left margin and output its highlight properties.

`option` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`option` (*CONDITION* `option-error`) [Method]

`option-sticky-distance` *OPTION NAMEARG* [Generic Function]
Try to match OPTION's short name with a sticky argument against NAMEARG.
If OPTION matches, return the length of OPTION's short name; otherwise 0.

Package [`net.didierverna.clon`], page 27

Source [`option`], page 13 (Lisp file)

Methods

`option-sticky-distance` (*OPTION* `valued-option`) *NAMEARG* [Method]
Try to match OPTION's short name with a sticky argument against NAMEARG.
If OPTION matches, return its short name's length; otherwise 0.

Source [`valued`], page 14 (Lisp file)

`option-sticky-distance` (*OPTION* `option`) *NAMEARG* [Method]
Return 0 (non-valued options don't take any argument, sticky or not).

`output-stream` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`output-stream` (*SHEET* `sheet`) [Method]
The sheet's output stream.

Source [`sheet`], page 21 (Lisp file)

`parent` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`parent` (*FACE* `face`) [Method]
The face parent.

Source [`face`], page 20 (Lisp file)

`path-type` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`path-type` (*PATH* `path`) [Method]
The path type.

Source [`path`], page 16 (Lisp file)

`postfix` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`postfix` (*CONTEXT* `context`) [Method]
Return the postfix of CONTEXT's synopsis.

Source [`context`], page 23 (Lisp file)

`postfix` (*SYNOPSIS* `synopsis`) [Method]
A postfix to the program synopsis.

Source [`synopsis`], page 19 (Lisp file)

`potential-pack` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`potential-pack` (*SYNOPSIS* `synopsis`) [Method]
The potential pack string.

Source [`synopsis`], page 19 (Lisp file)

`potential-pack-p` *PACK THERE* [Generic Function]
Return t if PACK is a potential pack in THERE.

Package [`net.didierverna.clon`], page 27

Source [`synopsis`], page 19 (Lisp file)

Methods

`potential-pack-p` *PACK* (*CONTEXT* `context`) [Method]
Return t if PACK (a string) is a potential pack in CONTEXT.

Source [`context`], page 23 (Lisp file)

`potential-pack-p` *PACK* (*SYNOPSIS* `synopsis`) [Method]
Return t if PACK is a potential pack for SYNOPSIS.

`print-help-spec` *SHEET HELP-SPEC* [Generic Function]
Print HELP-SPEC on SHEET.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Methods

`print-help-spec` *SHEET HELP-SPEC* `before` [Method]

`print-help-spec` *SHEET* (*CHAR* `character`) [Method]
Print CHAR on SHEET with the current face.

`print-help-spec` *SHEET* (*CHAR-VECTOR* `simple-vector`) [Method]
Print CHAR-VECTOR on SHEET with the current face.

`print-help-spec` *SHEET* (*STRING* `string`) [Method]
Print STRING on SHEET with the current face.

`print-help-spec` *SHEET* (*HELP-SPEC* `list`) [Method]
Open HELP-SPEC's face and print all of its items with it.

`retrieve-from-environment` *OPTION ENV-VAL* [Generic Function]
Retrieve OPTION's value from the environment.
ENV-VAL is the value stored in the associated environment variable.

Package [`net.didierverna.clon`], page 27

Source [`environ`], page 19 (Lisp file)

Methods

`retrieve-from-environment` *OPTION ENV-VAL* `before` [Method]
Assert that ENV-VAL is not null.

`retrieve-from-environment` (*FLAG* `flag`) *ENV-VAL* [Method]

`retrieve-from-environment` (*OPTION* `valued-option`) [Method]
ENV-VAL

`retrieve-from-long-call` *OPTION CMDLINE-NAME* **&optional** *CMDLINE-ARGUMENT CMDLINE* [Generic Function]

Retrieve OPTION's value from a long call.
CMDLINE-NAME is the name used on the command-line.
CMDLINE-ARGUMENT is a potentially already parsed cmdline argument.
Otherwise, CMDLINE is where to find an argument.
This function returns three values:
- the retrieved value,
- the value source,
- the new command-line (possibly with the first item popped if the option requires an argument).

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

Methods

`retrieve-from-long-call` (*OPTION* `option`) [Method]
CMDLINE-NAME **&optional** *CMDLINE-ARGUMENT CMDLINE*

`retrieve-from-long-call` (*OPTION* `valued-option`) [Method]
CMDLINE-NAME **&optional** *CMDLINE-ARGUMENT CMDLINE*

`retrieve-from-negated-call` *OPTION* [Generic Function]

Retrieve OPTION's value from a negated call.

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

Methods

`retrieve-from-negated-call` (*OPTION* `option`) [Method]

`retrieve-from-negated-call` (*OPTION* `valued-option`) [Method]

`retrieve-from-negated-call` (*NEGATABLE* `negatable`) [Method]

`retrieve-from-short-call` *OPTION* **&optional** *CMDLINE-ARGUMENT CMDLINE* [Generic Function]

Retrieve OPTION's value from a short call.
CMDLINE-ARGUMENT is a potentially already parsed cmdline argument.
Otherwise, CMDLINE is where to find an argument.
This function returns three values:
- the retrieved value,
- the value source,
- the new command-line (possibly with the first item popped if the option requires an argument).

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

Methods

`retrieve-from-short-call` (*OPTION* `option`) **&optional** [Method]
CMDLINE-ARGUMENT CMDLINE

`retrieve-from-short-call` (*OPTION* `valued-option`) [Method]
&optional *CMDLINE-ARGUMENT CMDLINE*

`right-padding` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`right-padding` (*FACE* `face`) [Method]
The face right padding.
This property can take the following forms:
- <NUMBER>: the padding is relative to the enclosing face,
- SELF: the padding is set to wherever the face happens to be closed,
- (<NUMBER> ABSOLUTE): the padding is set in absolute value,
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.

Source [`face`], page 20 (Lisp file)

`search-path` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`search-path` (*CONTEXT* `context`) [Method]
The search path for Clon files.

Source [`context`], page 23 (Lisp file)

`sface-tree` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`sface-tree` (*SHEET* `sheet`) [Method]
The sheet's sface tree.

Source [`sheet`], page 21 (Lisp file)

`short-call` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`short-call` (*CONDITION* [Method]
`unrecognized-short-call-error`)

`short-name` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`short-name` (*OPTION* `option`) [Method]
The option's short name.

Source [`option`], page 13 (Lisp file)

`short-pack` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`short-pack` (*CONTEXT* `context`) [Method]
Return the short pack of CONTEXT's synopsis.

Source [`context`], page 23 (Lisp file)

`short-pack` (*SYNOPSIS* `synopsis`) [Method]
The short pack string.

Source [`synopsis`], page 19 (Lisp file)

`short-pack-char` *OPTION* **&optional** *AS-STRING* [Generic Function]
Return OPTION's short pack character, if any.
If AS-STRING, return a string of that character.

Package [`net.didierverna.clon`], page 27

Source [`option`], page 13 (Lisp file)

Methods

`short-pack-char` (*OPTION* `valued-option`) **&optional** *AS-STRING* [Method]
Return OPTION's short pack character if OPTION's argument is optional.

Source [`valued`], page 14 (Lisp file)

`short-pack-char` (*OPTION* `option`) **&optional** *AS-STRING* [Method]
Return OPTION's potential pack character.

`short-syntax-help-spec-prefix` *OPTION* [Generic Function]
Return the help specification prefix for OPTION's short call.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

Methods

`short-syntax-help-spec-prefix` (*OPTION* `negatable`) [Method]

Source [`negatable`], page 15 (Lisp file)

`short-syntax-help-spec-prefix` (*OPTION* `valued-option`) [Method]

`sibling` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`sibling` (*SFACE* `sface`) [Method]
The SFace's raw sibling.

Source [`sheet`], page 21 (Lisp file)

`stream-ioctl-output-handle` *STREAM* [Generic Function]
Return STREAM's ioctl output handle or NIL.

Package [`net.didierverna.clon`], page 27

Source [`termio`], page 11 (Lisp file)

Methods

`stream-ioctl-output-handle` (*STREAM* `synonym-stream`) [Method]

`stream-ioctl-output-handle` (*STREAM* `two-way-stream`) [Method]

`stream-ioctl-output-handle` (*STREAM* `fd-stream`) [Method]

`stream-ioctl-output-handle` *STREAM* [Method]

`stringify` *VALUED-OPTION VALUE* [Generic Function]

Transform VALUED-OPTION's VALUE into an argument.
This is the opposite of argument conversion.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

Methods

`stringify` (*XSWITCH* `xswitch`) *VALUE* [Method]

Transform XSWITCH's VALUE into an argument.

Source [`xswitch`], page 17 (Lisp file)

`stringify` (*ENUM* `enum`) *VALUE* [Method]

Transform ENUM's VALUE into an argument.

Source [`enum`], page 17 (Lisp file)

`stringify` (*PATH* `path`) *VALUE* [Method]

Transform PATH's VALUE into an argument.

Source [`path`], page 16 (Lisp file)

`stringify` (*LISPOBJ* `lispobj`) *VALUE* [Method]

Transform LISPOBJ's VALUE into an argument.

Source [`lispobj`], page 16 (Lisp file)

`stringify` (*STROPT* `stropt`) *VALUE* [Method]

Transform STROPT's VALUE into an argument.

Source [`stropt`], page 16 (Lisp file)

`stringify` (*SWITCH* `switch`) *VALUE* [Method]

Transform SWITCH's VALUE into an argument.

Source [`switch`], page 15 (Lisp file)

`subface` *FACE name*(*s*) [Generic Function]

Return subface of FACE named NAME(S) or nil.
If a list of names is provided instead of a single one, follow a subface branch matching those names to find the leaf face.

Package [`net.didierverna.clon`], page 27

Source [`face`], page 20 (Lisp file)

Methods

`subface` *FACE* (*NAME* `symbol`) [Method]

Return FACE'subface named NAME, or nil.

`subface` *FACE* (*NAMES* `list`) **&aux** *BRANCH* [Method]

Return the leaf face from FACE'subbranch matching NAMES, or nil.

`subfaces` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`subfaces` (*FACE* `face`) [Method]
The face children.

Source [`face`], page 20 (Lisp file)

`synopsis` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`synopsis` (*CONTEXT* `context`) [Method]
The program synopsis.

Source [`context`], page 23 (Lisp file)

`theme` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`theme` (*CONTEXT* `context`) [Method]
The theme filename.

Source [`context`], page 23 (Lisp file)

`top-padding` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`top-padding` (*HELP-SPEC* `list`) [Method]

Source [`sheet`], page 21 (Lisp file)

`top-padding` *OTHER* [Method]

Source [`sheet`], page 21 (Lisp file)

`top-padding` (*FACE* `face`) [Method]
The face top padding.
This property can take the following forms:
- nil: the output can start right away,
- 0: the output should start on the next line,
- N>0: there should be N empty lines before the output.

Source [`face`], page 20 (Lisp file)

`traversedp` *OBJECT* [Generic Function]
`(setf traversedp)` *NEW-VALUE OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`traversedp` (*ITEM* `item`) [Method]
`(setf traversedp)` *NEW-VALUE* (*ITEM* `item`) [Method]
The item's traversal state.

Source [`item`], page 12 (Lisp file)

`typespec` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`typespec` (*LISPOBJ* `lispobj`) [Method]
A type specifier the option's value should satisfy.

Source [`lispobj`], page 16 (Lisp file)

`underline` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`underline` (*FACE* `face`) [Method]
The face's underline level.

Source [`face`], page 20 (Lisp file)

`untraverse` *ITEM* [Generic Function]
Reset ITEM's traversal state, and return ITEM.

Package [`net.didierverna.clon`], page 27

Source [`item`], page 12 (Lisp file)

Methods

`untraverse` (*CONTEXT* `context`) [Method]
Untraverse CONTEXT synopsis.

Source [`context`], page 23 (Lisp file)

`untraverse` (*CONTAINER* `container`) [Method]
Untraverse all CONTAINER items.

Source [`container`], page 18 (Lisp file)

`untraverse` (*OPTION* `option`) [Method]
OPTION is a terminal object: just return it.

Source [`option`], page 13 (Lisp file)

`untraverse` (*TEXT* `text`) [Method]
TEXT is a terminal object: just return it.

Source [`text`], page 13 (Lisp file)

`untraverse` (*ITEM* `item`) `after` [Method]
Mark ITEM as untraversed.

`value` *CONDITION* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`value` (*CONDITION* `invalid-value`) [Method]

`visiblep` *OBJECT* [Generic Function]

Package [`net.didierverna.clon`], page 27

Methods

`visiblep` (*FACE* `face`) [Method]
Whether the face is visible.

Source [face], page 20 (Lisp file)

`yes-values` *OBJECT* [Generic Function]
`(setf yes-values)` *NEW-VALUE OBJECT* [Generic Function]

Package [net.didierverna.clon], page 27

Methods

`yes-values` (*SWITCH-BASE* `switch-base`) [Method]
`(setf yes-values)` *NEW-VALUE* (*SWITCH-BASE* `switch-base`) [Method]
The possible 'yes' values.

Source [switch-base], page 15 (Lisp file)

6.2.6 Conditions

`cmdline-error` [Condition]
An error related to a command-line item.

Package [net.didierverna.clon], page 27

Source [cmdline], page 18 (Lisp file)

Direct superclasses
`error`

Direct subclasses
- [cmdline-option-error], page 92 (condition)
- [invalid-short-equal-syntax], page 95 (condition)
- [invalid-negated-equal-syntax], page 94 (condition)
- [cmdline-junk-error], page 91 (condition)
- [unrecognized-short-call-error], page 97 (condition)
- [unrecognized-negated-call-error], page 97 (condition)
- [unknown-cmdline-option-error], page 96 (condition)

Direct methods
[item], page 79 (method)

Direct slots

`item` [Slot]
The concerned command-line item.

Initargs `:item`

Readers [item], page 79 (generic function)

`cmdline-junk-error` [Condition]
An error related to a command-line piece of junk.

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

Direct superclasses
[cmdline-error], page 91 (condition)

Direct methods
[junk], page 79 (method)

Direct slots

`item` [Slot]
The piece of junk appearing on the command-line.

Initargs `:junk`, `:item`

Readers [junk], page 79 (generic function)

`cmdline-option-error` [Condition]
An error related to a command-line (known) option.

Package [net.didierverna.clon], page 27

Source [cmdline], page 18 (Lisp file)

Direct superclasses
- [option-error], page 95 (condition)
- [cmdline-error], page 91 (condition)

Direct subclasses
- [spurious-cmdline-argument], page 96 (condition)
- [invalid-negated-syntax], page 94 (condition)
- [invalid-cmdline-argument], page 94 (condition)
- [missing-cmdline-argument], page 95 (condition)

Direct methods
[name], page 81 (method)

Direct slots

`item` [Slot]
The option's name as it appears on the command-line.

Initargs `:name`, `:item`

Readers [name], page 81 (generic function)

`environment-error` [Condition]
An error related to an environment variable.

Package [net.didierverna.clon], page 27

Source [environ], page 19 (Lisp file)

Direct superclasses
`error`

Direct subclasses
[environmental-option-error], page 93 (condition)

Direct methods
[env-var], page 75 (method)

Direct slots

`env-var` [Slot]
The concerned environment variable.

Initargs `:env-var`

Readers [env-var], page 75 (generic function)

`environmental-option-error` [Condition]

An error related to an option's environment variable.

Package [`net.didierverna.clon`], page 27

Source [`environ`], page 19 (Lisp file)

Direct superclasses

- [`option-error`], page 95 (condition)
- [`environment-error`], page 92 (condition)

Direct subclasses

[`invalid-environment-value`], page 94 (condition)

`home-directory` [Condition]

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

Direct superclasses

`warning`

Direct methods

- `error-string`
- [`error-string`], page 76 (method)

Direct slots

`error-string` [Slot]

Initargs `:error-string`

Readers [`error-string`], page 76 (generic function)

Writers [`(setf error-string)`], page 76 (generic function)

`invalid-argument` [Condition]

An invalid argument error.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

Direct superclasses

[`option-error`], page 95 (condition)

Direct subclasses

- [`invalid-cmdline-argument`], page 94 (condition)
- [`invalid-environment-value`], page 94 (condition)

Direct methods

- [`comment`], page 73 (method)
- [`argument`], page 70 (method)

Direct slots

`argument` [Slot]

The invalid argument.

Initargs `:argument`

Readers [`argument`], page 70 (generic function)

`comment` [Slot]
An additional comment about the error.

Initargs `:comment`

Readers [`comment`], page 73 (generic function)

`invalid-cmdline-argument` [Condition]
An invalid command-line argument error.

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

Direct superclasses

- [`invalid-argument`], page 93 (condition)
- [`cmdline-option-error`], page 92 (condition)

`invalid-environment-value` [Condition]
An invalid environment variable's value error.

Package [`net.didierverna.clon`], page 27

Source [`environ`], page 19 (Lisp file)

Direct superclasses

- [`invalid-argument`], page 93 (condition)
- [`environmental-option-error`], page 93 (condition)

Direct methods

[`env-val`], page 75 (method)

Direct slots

`argument` [Slot]
The invalid environment variable value.

Initargs `:env-val`, `:argument`

Readers [`env-val`], page 75 (generic function)

`invalid-negated-equal-syntax` [Condition]
An error related to a negated-equal syntax.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

Direct superclasses

[`cmdline-error`], page 91 (condition)

`invalid-negated-syntax` [Condition]
An invalid negated syntax error.

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

Direct superclasses

[`cmdline-option-error`], page 92 (condition)

`invalid-short-equal-syntax` [Condition]
An error related to a short-equal syntax.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

Direct superclasses
[`cmdline-error`], page 91 (condition)

`invalid-value` [Condition]
An invalid value error.

Package [`net.didierverna.clon`], page 27

Source [`valued`], page 14 (Lisp file)

Direct superclasses
[`option-error`], page 95 (condition)

Direct methods
- [`comment`], page 73 (method)
- [`value`], page 90 (method)

Direct slots

`value` [Slot]
The invalid value.

Initargs `:value`

Readers [`value`], page 90 (generic function)

`comment` [Slot]
An additional comment about the error.

Initargs `:comment`

Readers [`comment`], page 73 (generic function)

`missing-cmdline-argument` [Condition]
A missing command-line argument error.

Package [`net.didierverna.clon`], page 27

Source [`cmdline`], page 18 (Lisp file)

Direct superclasses
[`cmdline-option-error`], page 92 (condition)

`option-error` [Condition]
An error related to an option.

Package [`net.didierverna.clon`], page 27

Source [`option`], page 13 (Lisp file)

Direct superclasses
`error`

Direct subclasses
- [`invalid-value`], page 95 (condition)
- [`invalid-argument`], page 93 (condition)
- [`cmdline-option-error`], page 92 (condition)

- [environmental-option-error], page 93 (condition)

Direct methods

[option], page 82 (method)

Direct slots

option [Slot]

The concerned option.

Initargs :option

Readers [option], page 82 (generic function)

spurious-cmdline-argument [Condition]

A spurious command-line argument error.

Package [net.didierverna.clon], page 27

Source [cmdline], page 18 (Lisp file)

Direct superclasses

[cmdline-option-error], page 92 (condition)

Direct methods

[argument], page 70 (method)

Direct slots

argument [Slot]

The spurious argument.

Initargs :argument

Readers [argument], page 70 (generic function)

unknown-cmdline-option-error [Condition]

An error related to an unknown command-line option.

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

Direct superclasses

[cmdline-error], page 91 (condition)

Direct methods

- [argument], page 70 (method)
- [name], page 81 (method)

Direct slots

item [Slot]

The option's name as it appears on the command-line.

Initargs :name, :item

Readers [name], page 81 (generic function)

argument [Slot]

The option's command-line argument.

Initargs :argument

Readers [argument], page 70 (generic function)

`unrecognized-negated-call-error` [Condition]
An error related to an unrecognized negated call.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

Direct superclasses
[`cmdline-error`], page 91 (condition)

Direct methods
[`negated-call`], page 81 (method)

Direct slots

`item` [Slot]
The unrecognized negated call on the command-line.

Initargs `:negated-call`, `:item`

Readers [`negated-call`], page 81 (generic function)

`unrecognized-short-call-error` [Condition]
An error related to an unrecognized short call.

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

Direct superclasses
[`cmdline-error`], page 91 (condition)

Direct methods
[`short-call`], page 86 (method)

Direct slots

`item` [Slot]
The unrecognized short call on the command-line.

Initargs `:short-call`, `:item`

Readers [`short-call`], page 86 (generic function)

6.2.7 Structures

`cmdline-option` [Structure]

Package [`net.didierverna.clon`], page 27

Source [`context`], page 23 (Lisp file)

Direct superclasses
`structure-object`

Direct slots

`name` [Slot]

Readers [`cmdline-option-name`], page 54 (function)

Writers [`(setf cmdline-option-name)`], page 54 (function)

`option` [Slot]

Readers [`cmdline-option-option`], page 54 (function)

Writers [`(setf cmdline-option-option)`], page 54 (function)

`value` [Slot]

Readers [`cmdline-option-value`], page 54 (function)

Writers [`(setf cmdline-option-value)`], page 54 (function)

`source` [Slot]

Readers [`cmdline-option-source`], page 54 (function)

Writers [`(setf cmdline-option-source)`], page 54 (function)

`frame` [Structure]

The FRAME structure.
This structure hold layout properties used for printing.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Direct superclasses

`structure-object`

Direct subclasses

[`highlight-frame`], page 98 (structure)

Direct methods

- [`close-frame`], page 73 (method)
- [`open-frame`], page 82 (method)

Direct slots

`sface` [Slot]

Readers [`frame-sface`], page 56 (function)

Writers [`(setf frame-sface)`], page 56 (function)

`left-margin` [Slot]

Readers [`frame-left-margin`], page 56 (function)

Writers [`(setf frame-left-margin)`], page 56 (function)

`right-margin` [Slot]

Readers [`frame-right-margin`], page 56 (function)

Writers [`(setf frame-right-margin)`], page 56 (function)

`highlight-frame` [Structure]

The HIGHLIGHT-FRAME structure.
This structure holds both layout and highlight properties used for printing.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Direct superclasses

[`frame`], page 98 (structure)

Direct methods

- [`close-frame`], page 73 (method)
- [`open-frame`], page 82 (method)

Direct slots

`highlight-property-instances` [Slot]

Readers [`highlight-frame-highlight-property-instances`], page 57 (function)

Writers [`(setf highlight-frame-highlight-property-instances)`], page 57 (function)

`highlight-property-instance` [Structure]

The HIGHLIGHT-PROPERTY-INSTANCE structure.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Direct superclasses

`structure-object`

Direct slots

`name` [Slot]

Readers [`highlight-property-instance-name`], page 58 (function)

Writers [`(setf highlight-property-instance-name)`], page 58 (function)

`value` [Slot]

Readers [`highlight-property-instance-value`], page 58 (function)

Writers [`(setf highlight-property-instance-value)`], page 58 (function)

6.2.8 Classes

`abstract-class` [Class]

The ABSTRACT-CLASS class.
This is the meta-class for abstract classes.

Package [`net.didierverna.clon`], page 27

Source [`util`], page 12 (Lisp file)

Direct superclasses

`standard-class`

Direct methods

- `validate-superclass`
- `validate-superclass`
- `make-instance`

`container` [Class]

The CONTAINER class.
This class is a mixin used in synopsis and groups to represent the program's command-line hierarchy.

Package [`net.didierverna.clon`], page 27

Source [`container`], page 18 (Lisp file)

Direct superclasses

[`item`], page 106 (class)

Direct subclasses

- [group], page 106 (class)
- [synopsis], page 113 (class)

Direct methods

- [mapoptions], page 81 (method)
- initialize-instance
- initialize-instance
- [help-spec], page 77 (method)
- [check-name-clash], page 72 (method)
- [check-name-clash], page 72 (method)
- [check-name-clash], page 72 (method)
- [untraverse], page 90 (method)
- [items], page 79 (method)

Direct slots

items [Slot]

The items in the container.

Type list

Initargs :items

Readers [items], page 79 (generic function)

context [Class]

The CONTEXT class.
This class represents the associatiion of a synopsis and a set of command-line options based on it.

Package [net.didierverna.clon], page 27

Source [context], page 23 (Lisp file)

Direct superclasses

standard-object

Direct methods

- initialize-instance
- [untraverse], page 90 (method)
- [mapoptions], page 81 (method)
- [potential-pack-p], page 84 (method)
- [clon-options-group], page 72 (method)
- [negated-pack], page 81 (method)
- [short-pack], page 87 (method)
- [postfix], page 83 (method)
- [error-handler], page 76 (method)
- [highlight], page 78 (method)
- [line-width], page 80 (method)
- [theme], page 89 (method)
- [search-path], page 86 (method)
- cmdline-options

- [cmdline-options], page 73 (method)
- [synopsis], page 89 (method)

Direct slots

`synopsis` [Slot]
The program synopsis.

Type `net.didierverna.clon::synopsis`

Initargs `:synopsis`

Initform `net.didierverna.clon:*synopsis*`

Readers [synopsis], page 89 (generic function)

`progname` [Slot]
The program name as it appears on the command-line.

Type `string`

`cmdline-options` [Slot]
The options from the command-line.

Type `list`

Readers [cmdline-options], page 73 (generic function)

Writers [(setf cmdline-options)], page 73 (generic function)

`remainder` [Slot]
The non-Clon part of the command-line.

Type `list`

`search-path` [Slot]
The search path for Clon files.

Readers [search-path], page 86 (generic function)

`theme` [Slot]
The theme filename.

Readers [theme], page 89 (generic function)

`line-width` [Slot]
The line width for help display.

Readers [line-width], page 80 (generic function)

`highlight` [Slot]
Clon's output highlight mode.

Readers [highlight], page 78 (generic function)

`error-handler` [Slot]
The behavior to adopt on option retrieval errors.

Type `symbol`

Initform `:quit`

Readers [error-handler], page 76 (generic function)

Direct Default Initargs

Initarg	Value
`:cmdline`	`(net.didierverna.clon:cmdline)`

`enum` [Class]

The ENUM class.
This class implements options whose values belong to a set of keywords.

Package [`net.didierverna.clon`], page 27

Source [`enum`], page 17 (Lisp file)

Direct superclasses

- [`valued-option`], page 114 (class)
- [`enum-base`], page 102 (class)

Direct methods

- [`convert`], page 74 (method)
- [`check`], page 71 (method)
- [`stringify`], page 88 (method)

Direct slots

`argument-name` [Slot]

Initform `"type"`

`enum-base` [Class]

The ENUM-BASE abstract class.
This class provides support for options including enumerated values.

Package [`net.didierverna.clon`], page 27

Source [`enum-base`], page 17 (Lisp file)

Direct superclasses

`standard-object`

Direct subclasses

- [`enum`], page 102 (class)
- [`xswitch`], page 116 (class)

Direct methods

- `initialize-instance`
- [`enum`], page 75 (method)

Direct slots

`enum` [Slot]

The set of possible values.

Initargs `:enum`

Readers [`enum`], page 75 (generic function)

`face` [Class]

The FACE class.

Package [`net.didierverna.clon`], page 27

Source [`face`], page 20 (Lisp file)

Direct superclasses

`standard-object`

Direct subclasses

[`sface`], page 110 (class)

Direct methods

- `initialize-instance`
- `initialize-instance`
- `initialize-instance`
- `slot-unbound`
- [`parent`], page 83 (method)
- [`subfaces`], page 89 (method)
- [`background`], page 71 (method)
- [`foreground`], page 76 (method)
- [`framedp`], page 76 (method)
- [`crossed-out-p`], page 75 (method)
- [`concealedp`], page 74 (method)
- [`inversep`], page 79 (method)
- [`blink`], page 71 (method)
- [`underline`], page 90 (method)
- [`italicp`], page 79 (method)
- [`intensity`], page 78 (method)
- [`item-separator`], page 79 (method)
- [`bottom-padding`], page 71 (method)
- [`top-padding`], page 89 (method)
- [`right-padding`], page 86 (method)
- [`left-padding`], page 80 (method)
- [`visiblep`], page 91 (method)
- [`name`], page 81 (method)

Direct slots

`name` [Slot]

The face name.

Initargs `:name`

Readers [`name`], page 81 (generic function)

`visiblep` [Slot]

Whether the face is visible.

Initargs `:visible`

Initform `t`

Readers [`visiblep`], page 90 (generic function)

`left-padding` [Slot]

The face left padding.
This property can take the following forms:
- <NUMBER>: the padding is relative to the enclosing face,

- SELF: the padding is set to wherever the face happens to be opened,
- (<NUMBER> ABSOLUTE): the padding is set in absolute value,
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.

Initargs `:padding-left`

Initform `0`

Readers [`left-padding`], page 79 (generic function)

`right-padding` [Slot]

The face right padding.
This property can take the following forms:
- <NUMBER>: the padding is relative to the enclosing face,
- SELF: the padding is set to wherever the face happens to be closed,
- (<NUMBER> ABSOLUTE): the padding is set in absolute value,
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.

Initargs `:padding-right`

Initform `'net.didierverna.clon::self`

Readers [`right-padding`], page 86 (generic function)

`top-padding` [Slot]

The face top padding.
This property can take the following forms:
- nil: the output can start right away,
- 0: the output should start on the next line,
- N>0: there should be N empty lines before the output.

Initargs `:padding-top`

Readers [`top-padding`], page 89 (generic function)

`bottom-padding` [Slot]

The face bottom padding.
This property can take the following forms:
- nil: the next output can start right at the end of this face's,
- 0: the next output should start on the next line,
- N>0: there should be N empty lines before the next output.

Initargs `:padding-bottom`

Readers [`bottom-padding`], page 71 (generic function)

`item-separator` [Slot]

The face item separator.

Initargs `:item-separator`

Initform `#\`

Readers [`item-separator`], page 79 (generic function)

`intensity` [Slot]

The face intensity.

Initargs `:intensity`

Readers [`intensity`], page 78 (generic function)

`italicp` [Slot]
The face's italic status.

Initargs `:italic`

Readers [`italicp`], page 79 (generic function)

`underline` [Slot]
The face's underline level.

Initargs `:underline`

Readers [`underline`], page 90 (generic function)

`blink` [Slot]
The face's blink speed.

Initargs `:blink`

Readers [`blink`], page 71 (generic function)

`inversep` [Slot]
The face's inverse video status.

Initargs `:inverse`

Readers [`inversep`], page 79 (generic function)

`concealedp` [Slot]
The face's concealed status.

Initargs `:concealed`

Readers [`concealedp`], page 73 (generic function)

`crossed-out-p` [Slot]
The face's crossed out status.

Initargs `:crossed-out`

Readers [`crossed-out-p`], page 75 (generic function)

`framedp` [Slot]
The face's framed status.

Initargs `:framed`

Readers [`framedp`], page 76 (generic function)

`foreground` [Slot]
The face foreground.

Initargs `:foreground`

Readers [`foreground`], page 76 (generic function)

`background` [Slot]
The face background.

Initargs `:background`

Readers [`background`], page 71 (generic function)

`subfaces` [Slot]
The face children.

Initargs `:subfaces`

Readers [`subfaces`], page 89 (generic function)

`parent` [Slot]
The face parent.

Readers [`parent`], page 83 (generic function)

`flag` [Class]
The FLAG class.
This class implements options that don't take any argument.

Package [`net.didierverna.clon`], page 27

Source [`flag`], page 14 (Lisp file)

Direct superclasses
[`option`], page 108 (class)

Direct methods
[`retrieve-from-environment`], page 84 (method)

`group` [Class]
The GROUP class.
This class groups other groups, options or strings together, effectively implementing hierarchical program command-line.

Package [`net.didierverna.clon`], page 27

Source [`group`], page 18 (Lisp file)

Direct superclasses
[`container`], page 99 (class)

Direct methods
- [`help-spec`], page 77 (method)
- [`header`], page 77 (method)

Direct slots

`header` [Slot]
The group's header.

Initargs `:header`

Readers [`header`], page 77 (generic function)

`item` [Class]
The ITEM class.
This class is the base class for all synopsis items.

Package [`net.didierverna.clon`], page 27

Source [`item`], page 12 (Lisp file)

Direct superclasses
`standard-object`

Direct subclasses
- [`text`], page 114 (class)

- [option], page 108 (class)
- [container], page 99 (class)

Direct methods

- [mapoptions], page 81 (method)
- [help-spec], page 78 (method)
- [untraverse], page 90 (method)
- [hiddenp], page 78 (method)
- traversedp
- [traversedp], page 89 (method)

Direct slots

`traversedp` [Slot]
The item's traversal state.

Readers [traversedp], page 89 (generic function)

Writers [(setf traversedp)], page 89 (generic function)

`hiddenp` [Slot]
Whether the item is hidden in help strings.

Initargs :hidden

Readers [hiddenp], page 78 (generic function)

`lispobj` [Class]
The LISPOBJ class.
This class implements read-from-string options.

Package [net.didierverna.clon], page 27

Source [lispobj], page 16 (Lisp file)

Direct superclasses
[valued-option], page 114 (class)

Direct methods

- [convert], page 74 (method)
- [check], page 71 (method)
- [stringify], page 88 (method)
- [typespec], page 90 (method)

Direct slots

`argument-name` [Slot]

Initform "obj"

`typespec` [Slot]
A type specifier the option's value should satisfy.

Initargs :typespec

Initform t

Readers [typespec], page 90 (generic function)

`negatable` [Class]

The NEGATABLE Class.
This class implements the negated syntax for the switch-based hierarchy.

Package [`net.didierverna.clon`], page 27

Source [`negatable`], page 15 (Lisp file)

Direct superclasses

`standard-object`

Direct subclasses

[`switch-base`], page 112 (class)

Direct methods

- [`retrieve-from-negated-call`], page 85 (method)
- [`negated-pack-char`], page 82 (method)
- [`short-syntax-help-spec-prefix`], page 87 (method)

`option` [Class]

The OPTION class.
This is the base class for all options.

Package [`net.didierverna.clon`], page 27

Source [`option`], page 13 (Lisp file)

Direct superclasses

[`item`], page 106 (class)

Direct subclasses

- [`flag`], page 106 (class)
- [`valued-option`], page 114 (class)

Direct methods

- [`mapoptions`], page 81 (method)
- [`retrieve-from-negated-call`], page 85 (method)
- [`retrieve-from-short-call`], page 85 (method)
- [`retrieve-from-long-call`], page 85 (method)
- `initialize-instance`
- `initialize-instance`
- [`negated-pack-char`], page 82 (method)
- [`short-pack-char`], page 87 (method)
- [`option-sticky-distance`], page 83 (method)
- [`check-name-clash`], page 72 (method)
- [`help-spec`], page 77 (method)
- [`untraverse`], page 90 (method)
- [`env-var`], page 75 (method)
- [`description`], page 75 (method)
- [`long-name`], page 80 (method)
- [`short-name`], page 86 (method)

Direct slots

`short-name` [Slot]

The option's short name.

Type `(or null string)`

Initargs `:short-name`

Readers [`short-name`], page 86 (generic function)

`long-name` [Slot]

The option's long name.

Type `(or null string)`

Initargs `:long-name`

Readers [`long-name`], page 80 (generic function)

`description` [Slot]

The option's description.

Type `(or null string)`

Initargs `:description`

Readers [`description`], page 75 (generic function)

`env-var` [Slot]

The option's associated environment variable.

Type `(or null string)`

Initargs `:env-var`

Readers [`env-var`], page 75 (generic function)

Direct Default Initargs

Initarg	Value
`:internal`	`nil`

`path` [Class]

The PATH class.

This class implements options whose values are (colon-separated lists of) pathnames.

Package [`net.didierverna.clon`], page 27

Source [`path`], page 16 (Lisp file)

Direct superclasses

[`valued-option`], page 114 (class)

Direct methods

- [`convert`], page 74 (method)
- [`check`], page 71 (method)
- [`stringify`], page 88 (method)
- [`path-type`], page 83 (method)

Direct slots

`argument-name` [Slot]

Initform `"path"`

`path-type` [Slot]

The path type.

Initargs `:type`

Readers [`path-type`], page 83 (generic function)

`sface` [Class]

The SFACE class.
An SFace is the association of a face and its raw sibling. The sibling is used to create subfaces which would be missing from the original, user defined one.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Direct superclasses
[`face`], page 102 (class)

Direct methods
[`sibling`], page 87 (method)

Direct slots

`sibling` [Slot]

The SFace's raw sibling.

Readers [`sibling`], page 87 (generic function)

`sheet` [Class]

The SHEET class.
This class implements the notion of sheet for printing Clon help.

Package [`net.didierverna.clon`], page 27

Source [`sheet`], page 21 (Lisp file)

Direct superclasses
`standard-object`

Direct methods

- `initialize-instance`
- `initialize-instance`
- `frames`
- [`frames`], page 76 (method)
- `column`
- [`column`], page 73 (method)
- [`sface-tree`], page 86 (method)
- [`highlightp`], page 78 (method)
- [`line-width`], page 80 (method)
- [`output-stream`], page 83 (method)

Direct slots

`output-stream` [Slot]

The sheet's output stream.

Type `stream`

Initargs `:output-stream`

Readers [`output-stream`], page 83 (generic function)

`line-width` [Slot]

The sheet's line width.

Type `(integer 1)`

Initargs `:line-width`

Readers [`line-width`], page 80 (generic function)

`highlightp` [Slot]
Whether to highlight SHEET's output.

Initargs `:highlightp`

Readers [`highlightp`], page 78 (generic function)

`sface-tree` [Slot]
The sheet's sface tree.

Readers [`sface-tree`], page 86 (generic function)

`column` [Slot]
The sheet's current column.

Type `(integer 0)`

Initform `0`

Readers [`column`], page 73 (generic function)

Writers [`(setf column)`], page 73 (generic function)

`frames` [Slot]
The stack of currently open frames.

Type `list`

Readers [`frames`], page 76 (generic function)

Writers [`(setf frames)`], page 76 (generic function)

`stropt` [Class]
The STROPT class.
This class implements options the values of which are strings.

Package [`net.didierverna.clon`], page 27

Source [`stropt`], page 16 (Lisp file)

Direct superclasses
[`valued-option`], page 114 (class)

Direct methods
- [`convert`], page 74 (method)
- [`check`], page 72 (method)
- [`stringify`], page 88 (method)

Direct slots

`argument-name` [Slot]

Initform `"str"`

`switch` [Class]
The SWITCH class.
This class implements boolean options.

Package [`net.didierverna.clon`], page 27

Source [`switch`], page 15 (Lisp file)

Direct superclasses

- [valued-option], page 114 (class)
- [switch-base], page 112 (class)

Direct methods

- initialize-instance
- [convert], page 74 (method)
- [check], page 72 (method)
- [stringify], page 88 (method)

switch-base [Class]

The SWITCH-BASE abstract class.
This class provides support for options including boolean values.

Package [net.didierverna.clon], page 27

Source [switch-base], page 15 (Lisp file)

Direct superclasses

[negatable], page 108 (class)

Direct subclasses

- [switch], page 111 (class)
- [xswitch], page 116 (class)

Direct methods

- initialize-instance
- initialize-instance
- [argument-style], page 70 (method)
- no-values
- [no-values], page 82 (method)
- yes-values
- [yes-values], page 91 (method)
- argument-styles
- [argument-styles], page 70 (method)

Direct slots

argument-styles [Slot]

The possible argument styles.
The position of every argument style in the list must correspond to the position of the associated strings in the yes-values and no-values slots.

Type list

Initargs :argument-styles

Readers [argument-styles], page 70 (generic function)

Writers [(setf argument-styles)], page 70 (generic function)

yes-values [Slot]

The possible 'yes' values.

Type list

Initargs :yes-values

Readers [yes-values], page 91 (generic function)

Writers [(setf yes-values)], page 91 (generic function)

`no-values` [Slot]
The possible 'no' values.

Type `list`

Initargs `:no-values`

Readers [`no-values`], page 82 (generic function)

Writers [`(setf no-values)`], page 82 (generic function)

`argument-style` [Slot]
The selected argument style.

Type `keyword`

Initargs `:argument-style`

Initform `:yes/no`

Readers [`argument-style`], page 70 (generic function)

Direct Default Initargs

Initarg	Value
`:argument-type`	`:optional`
`:argument-styles`	`'(:yes/no :on/off :true/false :yup/nope :yeah/nah)`
`:yes-values`	`'("yes" "on" "true" "yup" "yeah")`
`:no-values`	`'("no" "off" "false" "nope" "nah")`

`synopsis` [Class]
The SYNOPSIS class.
This class handles the description of the program's command-line options.

Package [`net.didierverna.clon`], page 27

Source [`synopsis`], page 19 (Lisp file)

Direct superclasses
[`container`], page 99 (class)

Direct methods
- `initialize-instance`
- `initialize-instance`
- [`potential-pack-p`], page 84 (method)
- [`help-spec`], page 77 (method)
- [`clon-options-group`], page 72 (method)
- [`potential-pack`], page 84 (method)
- [`negated-pack`], page 81 (method)
- [`short-pack`], page 87 (method)
- [`postfix`], page 83 (method)

Direct slots

`postfix` [Slot]
A postfix to the program synopsis.

Type `(or null string)`

Initargs `:postfix`

Readers [`postfix`], page 83 (generic function)

`short-pack` [Slot]
The short pack string.

Type `(or null string)`

Readers [`short-pack`], page 86 (generic function)

`negated-pack` [Slot]
The negated pack string.

Type `(or null string)`

Readers [`negated-pack`], page 81 (generic function)

`potential-pack` [Slot]
The potential pack string.

Type `(or null string)`

Readers [`potential-pack`], page 83 (generic function)

`clon-options-group` [Slot]
The Clon options group.

Type `net.didierverna.clon::group`

Initargs `:clon-options-group`

Readers [`clon-options-group`], page 72 (generic function)

`text` [Class]
The TEXT class.
This class implements plain text objects appearing in a synopsis.

Package [`net.didierverna.clon`], page 27

Source [`text`], page 13 (Lisp file)

Direct superclasses
[`item`], page 106 (class)

Direct methods
- [`check-name-clash`], page 72 (method)
- [`check-name-clash`], page 72 (method)
- [`help-spec`], page 77 (method)
- [`untraverse`], page 90 (method)
- [`contents`], page 74 (method)

Direct slots

`contents` [Slot]
The actual text string.

Type `string`

Initargs `:contents`

Readers [`contents`], page 74 (generic function)

`valued-option` [Class]
The VALUED-OPTION class.
This is the base class for options accepting arguments.

Package [net.didierverna.clon], page 27

Source [valued], page 14 (Lisp file)

Direct superclasses
[option], page 108 (class)

Direct subclasses
- [switch], page 111 (class)
- [stropt], page 111 (class)
- [lispobj], page 107 (class)
- [path], page 109 (class)
- [enum], page 102 (class)
- [xswitch], page 116 (class)

Direct methods
- [retrieve-from-environment], page 84 (method)
- [retrieve-from-negated-call], page 85 (method)
- [retrieve-from-short-call], page 85 (method)
- [retrieve-from-long-call], page 85 (method)
- initialize-instance
- initialize-instance
- [help-spec], page 77 (method)
- [short-syntax-help-spec-prefix], page 87 (method)
- [short-pack-char], page 87 (method)
- [option-sticky-distance], page 83 (method)
- [default-value], page 75 (method)
- [fallback-value], page 76 (method)
- [argument-required-p], page 70 (method)
- [argument-name], page 70 (method)

Direct slots

argument-name [Slot]
The option's argument display name.

Initargs :argument-name

Initform "arg"

Readers [argument-name], page 70 (generic function)

argument-required-p [Slot]
Whether the option's argument is required.

Readers [argument-required-p], page 70 (generic function)

fallback-value [Slot]
The option's fallback value.

Initargs :fallback-value

Readers [fallback-value], page 76 (generic function)

`default-value` [Slot]
The option's default value.

Initargs `:default-value`

Readers [`default-value`], page 75 (generic function)

Direct Default Initargs

Initarg	Value
`:argument-type`	`:required`

`xswitch` [Class]
The XSWITCH class.
This class merges the functionalities of switches and enumerations.
As such, the negated syntax is available for extended xswitches.

Package [`net.didierverna.clon`], page 27

Source [`xswitch`], page 17 (Lisp file)

Direct superclasses

- [`valued-option`], page 114 (class)
- [`enum-base`], page 102 (class)
- [`switch-base`], page 112 (class)

Direct methods

- [`convert`], page 74 (method)
- [`check`], page 71 (method)
- [`stringify`], page 88 (method)

Direct slots

`enum` [Slot]
The set of possible non-boolean values.

Appendix A Indexes

A.1 Concepts

C

core/meta.lisp 11
core/src/ 9
core/src/container.lisp 18
core/src/context.lisp 23
core/src/group.lisp 18
core/src/item.lisp 12
core/src/options/ 9
core/src/options/enum-base.lisp 17
core/src/options/enum.lisp 17
core/src/options/flag.lisp 14
core/src/options/lispobj.lisp 16
core/src/options/negatable.lisp 15
core/src/options/option.lisp 13
core/src/options/path.lisp 16
core/src/options/stropt.lisp 16
core/src/options/switch-base.lisp 15
core/src/options/switch.lisp 15
core/src/options/valued.lisp 14
core/src/options/xswitch.lisp 17
core/src/output/ 10
core/src/output/face.lisp 20
core/src/output/sheet.lisp 21
core/src/retrieval/ 10
core/src/retrieval/cmdline.lisp 18
core/src/retrieval/environ.lisp 19
core/src/synopsis.lisp 19
core/src/text.lisp 13
core/src/util.lisp 12

F

File, Lisp, core/meta.lisp 11
File, Lisp, core/src/container.lisp 18
File, Lisp, core/src/context.lisp 23
File, Lisp, core/src/group.lisp 18
File, Lisp, core/src/item.lisp 12
File, Lisp, core/src/options/enum-base.lisp ... 17
File, Lisp, core/src/options/enum.lisp 17
File, Lisp, core/src/options/flag.lisp 14
File, Lisp, core/src/options/lispobj.lisp 16
File, Lisp, core/src/options/negatable.lisp ... 15
File, Lisp, core/src/options/option.lisp 13
File, Lisp, core/src/options/path.lisp 16
File, Lisp, core/src/options/stropt.lisp 16
File, Lisp, core/src/options/switch-base.lisp 15
File, Lisp, core/src/options/switch.lisp 15
File, Lisp, core/src/options/valued.lisp 14
File, Lisp, core/src/options/xswitch.lisp 17
File, Lisp, core/src/output/face.lisp 20
File, Lisp, core/src/output/sheet.lisp 21
File, Lisp, core/src/retrieval/cmdline.lisp ... 18
File, Lisp, core/src/retrieval/environ.lisp ... 19
File, Lisp, core/src/synopsis.lisp 19
File, Lisp, core/src/text.lisp 13
File, Lisp, core/src/util.lisp 12
File, Lisp, net.didierverna.clon.asd 11
File, Lisp, net.didierverna.clon.core.asd 11
File, Lisp, net.didierverna.clon.setup.asd 11
File, Lisp, net.didierverna.clon.termio.asd ... 11
File, Lisp, setup/setup.lisp 25
File, Lisp, termio/sbcl/constants.lisp 11
File, Lisp, termio/termio.lisp 11

L

Lisp File, core/meta.lisp 11
Lisp File, core/src/container.lisp 18
Lisp File, core/src/context.lisp 23
Lisp File, core/src/group.lisp 18
Lisp File, core/src/item.lisp 12
Lisp File, core/src/options/enum-base.lisp 17
Lisp File, core/src/options/enum.lisp 17
Lisp File, core/src/options/flag.lisp 14
Lisp File, core/src/options/lispobj.lisp 16
Lisp File, core/src/options/negatable.lisp 15
Lisp File, core/src/options/option.lisp 13
Lisp File, core/src/options/path.lisp 16
Lisp File, core/src/options/stropt.lisp 16
Lisp File, core/src/options/switch-base.lisp .. 15
Lisp File, core/src/options/switch.lisp 15
Lisp File, core/src/options/valued.lisp 14
Lisp File, core/src/options/xswitch.lisp 17
Lisp File, core/src/output/face.lisp 20
Lisp File, core/src/output/sheet.lisp 21
Lisp File, core/src/retrieval/cmdline.lisp 18
Lisp File, core/src/retrieval/environ.lisp 19
Lisp File, core/src/synopsis.lisp 19
Lisp File, core/src/text.lisp 13
Lisp File, core/src/util.lisp 12
Lisp File, net.didierverna.clon.asd 11
Lisp File, net.didierverna.clon.core.asd 11
Lisp File, net.didierverna.clon.setup.asd 11
Lisp File, net.didierverna.clon.termio.asd 11
Lisp File, setup/setup.lisp 25
Lisp File, termio/sbcl/constants.lisp 11
Lisp File, termio/termio.lisp 11

M

Module, core/src/ 9
Module, core/src/options/ 9
Module, core/src/output/ 10
Module, core/src/retrieval/ 10

N

net.didierverna.clon.asd 11
net.didierverna.clon.core.asd 11
net.didierverna.clon.setup.asd 11
net.didierverna.clon.termio.asd 11

S

setup/setup.lisp 25

T

termio/sbcl/constants.lisp 11
termio/termio.lisp 11

A.2 Functions

%

%defgroup 50
%version 52

(

(setf argument-styles) 70
(setf cmdline-option-name) 54
(setf cmdline-option-option) 54
(setf cmdline-option-source) 54
(setf cmdline-option-value) 54
(setf cmdline-options) 73
(setf column) 73
(setf error-string) 76
(setf frame-left-margin) 56
(setf frame-right-margin) 56
(setf frame-sface) 56
(setf frames) 76
(setf highlight-frame-highlight-property-instances) 57
(setf highlight-frame-left-margin) 57
(setf highlight-frame-right-margin) 57
(setf highlight-frame-sface) 57
(setf highlight-property-instance-name) 58
(setf highlight-property-instance-value) 58
(setf no-values) 82
(setf traversedp) 89
(setf winsize-ws-col) 69
(setf winsize-ws-row) 69
(setf winsize-ws-xpixel) 69
(setf winsize-ws-ypixel) 70
(setf yes-values) 91

A

accumulate 50
add-subface 52
allocate-winsize 53
argument 70
argument-name 70
argument-popable-p 53
argument-required-p 70
argument-style 70
argument-styles 70
attach-face-tree 53
available-right-margin 53

B

background 71
beginning-of-string-p 53
blink 71
bottom-padding 71

C

check 71, 72
check-name-clash 72
clindent 53
clon-options-group 72
close-frame 73
close-line 53
close-sface 53
closest-match 54
cmdline 43
cmdline-convert 54
cmdline-option-name 54
cmdline-option-option 54
cmdline-option-p 54
cmdline-option-source 54
cmdline-option-value 54
cmdline-options 73
cmdline-options-p 43
cmdline-p 43
column 73
comment 73
complete-string 54
concealedp 73, 74
configuration 43
configure 44
contents 74
convert 74
copy-cmdline-option 55
copy-frame 55
copy-highlight-frame 55
copy-highlight-property-instance 55
copy-instance 74, 75
crossed-out-p 75
current-frame 55
current-left-margin 55
current-right-margin 55
current-sface 55

D

declare-valid-superclass 50
defabstract 50
default-value 75
defgroup 42
defindent 50
defoption 51
defsynopsis 42
description 75
directory-pathname-p 55
do-cmdline-options 42
do-options 51
dump 42

E

econd 51
endpush 51
enum 75
env-val 75
env-var 75
environment-convert 55
error-handler 76
error-string 76
exit 44
exit-abnormally 56

F

face-highlight-property-set-p 56
face-highlight-property-value 56
fallback-value 76
find-sface 56
flush-sheet 56
foreground 76
frame-left-margin 56
frame-p 56
frame-right-margin 56
frame-sface 56
framedp 76
frames 76
free-winsize 57
Function, %version 52
Function, (setf cmdline-option-name) 54
Function, (setf cmdline-option-option) 54
Function, (setf cmdline-option-source) 54
Function, (setf cmdline-option-value) 54
Function, (setf frame-left-margin) 56
Function, (setf frame-right-margin) 56
Function, (setf frame-sface) 56
Function, (setf highlight-frame-highlight-property-instances) 57
Function, (setf highlight-frame-left-margin) 57
Function, (setf highlight-frame-right-margin) 57
Function, (setf highlight-frame-sface) 57
Function, (setf highlight-property-instance-name) 58
Function, (setf highlight-property-instance-value) 58
Function, (setf winsize-ws-col) 69
Function, (setf winsize-ws-row) 69
Function, (setf winsize-ws-xpixel) 69
Function, (setf winsize-ws-ypixel) 70
Function, add-subface 52
Function, allocate-winsize 53
Function, argument-popable-p 53
Function, attach-face-tree 53
Function, available-right-margin 53
Function, beginning-of-string-p 53
Function, clindent 53
Function, close-line 53
Function, close-sface 53
Function, closest-match 54
Function, cmdline 43
Function, cmdline-convert 54
Function, cmdline-option-name 54
Function, cmdline-option-option 54
Function, cmdline-option-p 54
Function, cmdline-option-source 54
Function, cmdline-option-value 54
Function, cmdline-options-p 43
Function, cmdline-p 43
Function, complete-string 54
Function, configuration 43
Function, configure 44
Function, copy-cmdline-option 55
Function, copy-frame 55
Function, copy-highlight-frame 55
Function, copy-highlight-property-instance 55
Function, current-frame 55
Function, current-left-margin 55
Function, current-right-margin 55
Function, current-sface 55
Function, directory-pathname-p 55
Function, environment-convert 55
Function, exit 44
Function, exit-abnormally 56
Function, face-highlight-property-set-p 56
Function, face-highlight-property-value 56
Function, find-sface 56
Function, flush-sheet 56
Function, frame-left-margin 56
Function, frame-p 56
Function, frame-right-margin 56
Function, frame-sface 56
Function, free-winsize 57
Function, get-top-padding 57
Function, getenv 57
Function, getopt 44
Function, getopt-cmdline 44
Function, help 44
Function, help-spec-items-will-print 57
Function, highlight-frame-highlight-property-instances 57
Function, highlight-frame-left-margin 57
Function, highlight-frame-p 57
Function, highlight-frame-right-margin 57
Function, highlight-frame-sface 57
Function, highlight-property-instance-escape-sequence 57
Function, highlight-property-instance-name 58
Function, highlight-property-instance-p 58
Function, highlight-property-instance-value 58
Function, home-directory 58
Function, i-reader 58
Function, list-to-string 58
Function, macosp 58
Function, make-cmdline-option 58
Function, make-context 44
Function, make-enum 45
Function, make-flag 45
Function, make-frame 58
Function, make-group 45
Function, make-highlight-frame 59
Function, make-highlight-property-instance 59
Function, make-internal-enum 59
Function, make-internal-flag 59
Function, make-internal-lispobj 59
Function, make-internal-path 60
Function, make-internal-stropt 60
Function, make-internal-switch 61
Function, make-internal-text 61
Function, make-internal-xswitch 61
Function, make-lispobj 45
Function, make-path 46
Function, make-raw-face-tree 61
Function, make-raw-sface 62
Function, make-sheet 62
Function, make-stropt 46
Function, make-switch 47
Function, make-synopsis 47
Function, make-text 47
Function, make-xswitch 48
Function, match-option 62

Function, nickname-package 48
Function, open-line 62
Function, open-next-line 62
Function, open-sface 62
Function, option-abbreviation-distance 62
Function, option-call-p 62
Function, parent-generation 63
Function, pathname-component-null-p 63
Function, pop-frame 63
Function, potential-pack-char 63
Function, princ-char 63
Function, princ-highlight-property-instances 63
Function, princ-spaces 63
Function, princ-string 63
Function, print-error 64
Function, print-faced-help-spec 64
Function, print-help 64
Function, print-string 64
Function, progname 48
Function, push-frame 64
Function, putenv 64
Function, reach-column 64
Function, read-argument 64
Function, read-call 65
Function, read-env-val 65
Function, read-long-name 65
Function, read-sface-tree 65
Function, read-value 65
Function, release-status-number 65
Function, remainder 48
Function, remove-keys 65
Function, replace-key 65
Function, replace-keys 66
Function, restart-on-error 66
Function, restartable-check 66
Function, restartable-cmdline-convert 66
Function, restartable-cmdline-junk-error 66
Function, restartable-convert 67
Function, restartable-environment-convert 67
Function, restrict-because 67
Function, safe-left-margin 67
Function, safe-right-margin 67
Function, search-branch 67
Function, search-face 68
Function, search-option 68
Function, search-option-by-abbreviation 68
Function, search-option-by-name 68
Function, search-sticky-option 68
Function, select-keys 69
Function, setup-termio 48
Function, split-path 69
Function, stream-line-width 69
Function, tilde-reader 69
Function, try-read-sface-tree 69
Function, try-read-theme 69
Function, version 48
Function, winsize-ws-col 69
Function, winsize-ws-row 69
Function, winsize-ws-xpixel 69
Function, winsize-ws-ypixel 70

G

Generic Function, (setf argument-styles) 70
Generic Function, (setf cmdline-options) 73
Generic Function, (setf column) 73
Generic Function, (setf error-string) 76
Generic Function, (setf frames) 76
Generic Function, (setf no-values) 82
Generic Function, (setf traversedp) 89
Generic Function, (setf yes-values) 91
Generic Function, argument 70
Generic Function, argument-name 70
Generic Function, argument-required-p 70
Generic Function, argument-style 70
Generic Function, argument-styles 70
Generic Function, background 71
Generic Function, blink 71
Generic Function, bottom-padding 71
Generic Function, check 71
Generic Function, check-name-clash 72
Generic Function, clon-options-group 72
Generic Function, close-frame 73
Generic Function, cmdline-options 73
Generic Function, column 73
Generic Function, comment 73
Generic Function, concealedp 73
Generic Function, contents 74
Generic Function, convert 74
Generic Function, copy-instance 74
Generic Function, crossed-out-p 75
Generic Function, default-value 75
Generic Function, description 75
Generic Function, enum 75
Generic Function, env-val 75
Generic Function, env-var 75
Generic Function, error-handler 76
Generic Function, error-string 76
Generic Function, fallback-value 76
Generic Function, foreground 76
Generic Function, framedp 76
Generic Function, frames 76
Generic Function, get-bottom-padding 77
Generic Function, header 77
Generic Function, help-spec 77
Generic Function, help-spec-will-print 78
Generic Function, hiddenp 78
Generic Function, highlight 78
Generic Function, highlightp 78
Generic Function, intensity 78
Generic Function, inversep 79
Generic Function, italicp 79
Generic Function, item 79
Generic Function, item-separator 79
Generic Function, items 79
Generic Function, junk 79
Generic Function, left-padding 79
Generic Function, line-width 80
Generic Function, long-name 80
Generic Function, make-face-tree 80
Generic Function, mapoptions 80
Generic Function, name 81
Generic Function, negated-call 81
Generic Function, negated-pack 81
Generic Function, negated-pack-char 81
Generic Function, no-values 82

Generic Function, open-frame ... 82
Generic Function, option ... 82
Generic Function, option-sticky-distance ... 82
Generic Function, output-stream ... 83
Generic Function, parent ... 83
Generic Function, path-type ... 83
Generic Function, postfix ... 83
Generic Function, potential-pack ... 83
Generic Function, potential-pack-p ... 84
Generic Function, print-help-spec ... 84
Generic Function, retrieve-from-environment ... 84
Generic Function, retrieve-from-long-call ... 85
Generic Function, retrieve-from-negated-call ... 85
Generic Function, retrieve-from-short-call ... 85
Generic Function, right-padding ... 86
Generic Function, search-path ... 86
Generic Function, sface-tree ... 86
Generic Function, short-call ... 86
Generic Function, short-name ... 86
Generic Function, short-pack ... 86
Generic Function, short-pack-char ... 87
Generic Function, short-syntax-help-spec-prefix ... 87
Generic Function, sibling ... 87
Generic Function, stream-ioctl-output-handle ... 87
Generic Function, stringify ... 88
Generic Function, subface ... 88
Generic Function, subfaces ... 89
Generic Function, synopsis ... 89
Generic Function, theme ... 89
Generic Function, top-padding ... 89
Generic Function, traversedp ... 89
Generic Function, typespec ... 90
Generic Function, underline ... 90
Generic Function, untraverse ... 90
Generic Function, value ... 90
Generic Function, visiblep ... 90
Generic Function, yes-values ... 91
get-bottom-padding ... 77
get-top-padding ... 57
getenv ... 57
getopt ... 44
getopt-cmdline ... 44

H

header ... 77
help ... 44
help-spec ... 77, 78
help-spec-items-will-print ... 57
help-spec-will-print ... 78
hiddenp ... 78
highlight ... 78
highlight-frame-highlight-property-instances ... 57
highlight-frame-left-margin ... 57
highlight-frame-p ... 57
highlight-frame-right-margin ... 57
highlight-frame-sface ... 57
highlight-property-ecase ... 51
highlight-property-instance-escape-sequence ... 57
highlight-property-instance-name ... 58
highlight-property-instance-p ... 58
highlight-property-instance-value ... 58
highlightp ... 78
home-directory ... 58

I

i-reader ... 58
in-readtable ... 51
intensity ... 78
inversep ... 79
italicp ... 79
item ... 79
item-separator ... 79
items ... 79

J

junk ... 79

L

left-padding ... 79, 80
line-width ... 80
list-to-string ... 58
long-name ... 80

M

macosp ... 58
Macro, %defgroup ... 50
Macro, accumulate ... 50
Macro, declare-valid-superclass ... 50
Macro, defabstract ... 50
Macro, defgroup ... 42
Macro, defindent ... 50
Macro, defoption ... 51
Macro, defsynopsis ... 42
Macro, do-cmdline-options ... 42
Macro, do-options ... 51
Macro, dump ... 42
Macro, econd ... 51
Macro, endpush ... 51
Macro, highlight-property-ecase ... 51
Macro, in-readtable ... 51
Macro, map-frames ... 51
Macro, maybe-pop-argument ... 51
Macro, maybe-push ... 52
Macro, multiple-value-getopt-cmdline ... 43
Macro, replace-in-keys ... 52
Macro, restartable-invalid-negated-syntax-error ... 52
Macro, restartable-spurious-cmdline-argument-error ... 52
Macro, with-context ... 43
Macro, with-context-error-handler ... 52
Macro, with-winsize ... 52
make-cmdline-option ... 58
make-context ... 44
make-enum ... 45
make-face-tree ... 80
make-flag ... 45

make-frame 58
make-group 45
make-highlight-frame 59
make-highlight-property-instance 59
make-internal-enum 59
make-internal-flag 59
make-internal-lispobj 59
make-internal-path 60
make-internal-stropt 60
make-internal-switch 61
make-internal-text 61
make-internal-xswitch 61
make-lispobj 45
make-path 46
make-raw-face-tree 61
make-raw-sface 62
make-sheet 62
make-stropt 46
make-switch 47
make-synopsis 47
make-text 47
make-xswitch 48
map-frames 51
mapoptions 80, 81
match-option 62
maybe-pop-argument 51
maybe-push 52
Method, (setf argument-styles) 70
Method, (setf cmdline-options) 73
Method, (setf column) 73
Method, (setf error-string) 76
Method, (setf frames) 76
Method, (setf no-values) 82
Method, (setf traversedp) 89
Method, (setf yes-values) 91
Method, argument 70
Method, argument-name 70
Method, argument-required-p 70
Method, argument-style 70
Method, argument-styles 70
Method, background 71
Method, blink 71
Method, bottom-padding 71
Method, check 71, 72
Method, check-name-clash 72
Method, clon-options-group 72
Method, close-frame 73
Method, cmdline-options 73
Method, column 73
Method, comment 73
Method, concealedp 74
Method, contents 74
Method, convert 74
Method, copy-instance 75
Method, crossed-out-p 75
Method, default-value 75
Method, description 75
Method, enum 75
Method, env-val 75
Method, env-var 75
Method, error-handler 76
Method, error-string 76
Method, fallback-value 76
Method, foreground 76
Method, framedp 76
Method, frames 76
Method, get-bottom-padding 77
Method, header 77
Method, help-spec 77, 78
Method, help-spec-will-print 78
Method, hiddenp 78
Method, highlight 78
Method, highlightp 78
Method, intensity 78
Method, inversep 79
Method, italicp 79
Method, item 79
Method, item-separator 79
Method, items 79
Method, junk 79
Method, left-padding 80
Method, line-width 80
Method, long-name 80
Method, make-face-tree 80
Method, mapoptions 81
Method, name 81
Method, negated-call 81
Method, negated-pack 81
Method, negated-pack-char 82
Method, no-values 82
Method, open-frame 82
Method, option 82
Method, option-sticky-distance 83
Method, output-stream 83
Method, parent 83
Method, path-type 83
Method, postfix 83
Method, potential-pack 84
Method, potential-pack-p 84
Method, print-help-spec 84
Method, retrieve-from-environment 84
Method, retrieve-from-long-call 85
Method, retrieve-from-negated-call 85
Method, retrieve-from-short-call 85
Method, right-padding 86
Method, search-path 86
Method, sface-tree 86
Method, short-call 86
Method, short-name 86
Method, short-pack 87
Method, short-pack-char 87
Method, short-syntax-help-spec-prefix 87
Method, sibling 87
Method, stream-ioctl-output-handle 88
Method, stringify 88
Method, subface 88
Method, subfaces 89
Method, synopsis 89
Method, theme 89
Method, top-padding 89
Method, traversedp 89
Method, typespec 90
Method, underline 90
Method, untraverse 90
Method, value 90
Method, visiblep 91
Method, yes-values 91
multiple-value-getopt-cmdline 43

N

name ... 81
negated-call ... 81
negated-pack ... 81
negated-pack-char ... 81, 82
nickname-package ... 48
no-values ... 82

O

open-frame ... 82
open-line ... 62
open-next-line ... 62
open-sface ... 62
option ... 82
option-abbreviation-distance ... 62
option-call-p ... 62
option-sticky-distance ... 82, 83
output-stream ... 83

P

parent ... 83
parent-generation ... 63
path-type ... 83
pathname-component-null-p ... 63
pop-frame ... 63
postfix ... 83
potential-pack ... 83, 84
potential-pack-char ... 63
potential-pack-p ... 84
princ-char ... 63
princ-highlight-property-instances ... 63
princ-spaces ... 63
princ-string ... 63
print-error ... 64
print-faced-help-spec ... 64
print-help ... 64
print-help-spec ... 84
print-string ... 64
progname ... 48
push-frame ... 64
putenv ... 64

R

reach-column ... 64
read-argument ... 64
read-call ... 65
read-env-val ... 65
read-long-name ... 65
read-sface-tree ... 65
read-value ... 65
release-status-number ... 65
remainder ... 48
remove-keys ... 65
replace-in-keys ... 52
replace-key ... 65
replace-keys ... 66
restart-on-error ... 66
restartable-check ... 66
restartable-cmdline-convert ... 66
restartable-cmdline-junk-error ... 66
restartable-convert ... 67
restartable-environment-convert ... 67
restartable-invalid-negated-syntax-error ... 52
restartable-spurious-cmdline-argument-error ... 52
restrict-because ... 67
retrieve-from-environment ... 84
retrieve-from-long-call ... 85
retrieve-from-negated-call ... 85
retrieve-from-short-call ... 85
right-padding ... 86

S

safe-left-margin ... 67
safe-right-margin ... 67
search-branch ... 67
search-face ... 68
search-option ... 68
search-option-by-abbreviation ... 68
search-option-by-name ... 68
search-path ... 86
search-sticky-option ... 68
select-keys ... 69
setup-termio ... 48
sface-tree ... 86
short-call ... 86
short-name ... 86
short-pack ... 86, 87
short-pack-char ... 87
short-syntax-help-spec-prefix ... 87
sibling ... 87
split-path ... 69
stream-ioctl-output-handle ... 87, 88
stream-line-width ... 69
stringify ... 88
subface ... 88
subfaces ... 89
synopsis ... 89

T

theme ... 89
tilde-reader ... 69
top-padding ... 89
traversedp ... 89
try-read-sface-tree ... 69
try-read-theme ... 69
typespec ... 90

U

underline ... 90
untraverse ... 90

V

value ... 90
version ... 48
visiblep ... 90, 91

W

winsize-ws-col ... 69
winsize-ws-row ... 69

winsize-ws-xpixel 69
winsize-ws-ypixel 70
with-context 43
with-context-error-handler 52
with-winsize 52

Y

yes-values 91

A.3 Variables

*

context 41
executablep 49
highlight-properties 50
item-names 50
readtable 50
release-major-level 41
release-minor-level 41
release-name 41
release-status 41
release-status-level 41
synopsis 41

+

+tiocgwinsz+ 49

A

argument 93, 94, 96
argument-name 102, 107, 109, 111, 115
argument-required-p 115
argument-style 113
argument-styles 112

B

background 105
blink 105
bottom-padding 104

C

clon-options-group 114
cmdline-options 101
column 111
comment 94, 95
concealedp 105
configuration 42
Constant, +tiocgwinsz+ 49
Constant, offset-of-winsize-ws-col 49
Constant, offset-of-winsize-ws-row 49
Constant, offset-of-winsize-ws-xpixel 49
Constant, offset-of-winsize-ws-ypixel 49
Constant, size-of-winsize 49
contents 114
crossed-out-p 105

D

default-value 116
description 109

E

enum 102, 116
env-var 92, 109
error-handler 101
error-string 93

F

fallback-value 115
foreground 105
framedp 105
frames 111

H

header 106
hiddenp 107
highlight 101
highlight-property-instances 99
highlightp 111

I

intensity 104
inversep 105
italicp 105
item 91, 92, 96, 97
item-separator 104
items 100

L

left-margin 98
left-padding 103
line-width 101, 110
long-name 109

N

name 97, 99, 103
negated-pack 114
no-values 113

O

offset-of-winsize-ws-col 49
offset-of-winsize-ws-row 49
offset-of-winsize-ws-xpixel 49
offset-of-winsize-ws-ypixel 49
option 96, 97
output-stream 110

P

parent 106
path-type 109
postfix 113
potential-pack 114
progname 101

R

remainder 101
right-margin 98
right-padding 104

S

search-path ... 101
sface ... 98
sface-tree ... 111
short-name ... 108
short-pack ... 114
sibling ... 110
size-of-winsize ... 49
Slot, argument ... 93, 94, 96
Slot, argument-name ... 102, 107, 109, 111, 115
Slot, argument-required-p ... 115
Slot, argument-style ... 113
Slot, argument-styles ... 112
Slot, background ... 105
Slot, blink ... 105
Slot, bottom-padding ... 104
Slot, clon-options-group ... 114
Slot, cmdline-options ... 101
Slot, column ... 111
Slot, comment ... 94, 95
Slot, concealedp ... 105
Slot, contents ... 114
Slot, crossed-out-p ... 105
Slot, default-value ... 116
Slot, description ... 109
Slot, enum ... 102, 116
Slot, env-var ... 92, 109
Slot, error-handler ... 101
Slot, error-string ... 93
Slot, fallback-value ... 115
Slot, foreground ... 105
Slot, framedp ... 105
Slot, frames ... 111
Slot, header ... 106
Slot, hiddenp ... 107
Slot, highlight ... 101
Slot, highlight-property-instances ... 99
Slot, highlightp ... 111
Slot, intensity ... 104
Slot, inversep ... 105
Slot, italicp ... 105
Slot, item ... 91, 92, 96, 97
Slot, item-separator ... 104
Slot, items ... 100
Slot, left-margin ... 98
Slot, left-padding ... 103
Slot, line-width ... 101, 110
Slot, long-name ... 109
Slot, name ... 97, 99, 103
Slot, negated-pack ... 114
Slot, no-values ... 113
Slot, option ... 96, 97
Slot, output-stream ... 110
Slot, parent ... 106
Slot, path-type ... 109
Slot, postfix ... 113
Slot, potential-pack ... 114
Slot, progname ... 101
Slot, remainder ... 101
Slot, right-margin ... 98
Slot, right-padding ... 104
Slot, search-path ... 101
Slot, sface ... 98
Slot, sface-tree ... 111
Slot, short-name ... 108
Slot, short-pack ... 114
Slot, sibling ... 110
Slot, source ... 98
Slot, subfaces ... 106
Slot, synopsis ... 101
Slot, theme ... 101
Slot, top-padding ... 104
Slot, traversedp ... 107
Slot, typespec ... 107
Slot, underline ... 105
Slot, value ... 95, 98, 99
Slot, visiblep ... 103
Slot, yes-values ... 112
source ... 98
Special Variable, *context* ... 41
Special Variable, *executablep* ... 49
Special Variable, *highlight-properties* ... 50
Special Variable, *item-names* ... 50
Special Variable, *readtable* ... 50
Special Variable, *release-major-level* ... 41
Special Variable, *release-minor-level* ... 41
Special Variable, *release-name* ... 41
Special Variable, *release-status* ... 41
Special Variable, *release-status-level* ... 41
Special Variable, *synopsis* ... 41
Special Variable, configuration ... 42
subfaces ... 106
synopsis ... 101

T

theme ... 101
top-padding ... 104
traversedp ... 107
typespec ... 107

U

underline ... 105

V

value ... 95, 98, 99
visiblep ... 103

Y

yes-values ... 112

A.4 Data types

A

abstract-class 99

C

Class, abstract-class 99
Class, container 99
Class, context 100
Class, enum 102
Class, enum-base 102
Class, face 102
Class, flag 106
Class, group 106
Class, item 106
Class, lispobj 107
Class, negatable 108
Class, option 108
Class, path 109
Class, sface 110
Class, sheet 110
Class, stropt 111
Class, switch 111
Class, switch-base 112
Class, synopsis 113
Class, text 114
Class, valued-option 114
Class, xswitch 116
cmdline-error 91
cmdline-junk-error 91
cmdline-option 97
cmdline-option-error 92
Condition, cmdline-error 91
Condition, cmdline-junk-error 91
Condition, cmdline-option-error 92
Condition, environment-error 92
Condition, environmental-option-error 93
Condition, home-directory 93
Condition, invalid-argument 93
Condition, invalid-cmdline-argument 94
Condition, invalid-environment-value 94
Condition, invalid-negated-equal-syntax 94
Condition, invalid-negated-syntax 94
Condition, invalid-short-equal-syntax 95
Condition, invalid-value 95
Condition, missing-cmdline-argument 95
Condition, option-error 95
Condition, spurious-cmdline-argument 96
Condition, unknown-cmdline-option-error 96
Condition, unrecognized-negated-call-error ... 97
Condition, unrecognized-short-call-error 97
container 99
context 100

E

enum 102
enum-base 102
environment-error 92
environmental-option-error 93

F

face 102
flag 106
frame 98

G

group 106

H

highlight-frame 98
highlight-property-instance 99
home-directory 93

I

invalid-argument 93
invalid-cmdline-argument 94
invalid-environment-value 94
invalid-negated-equal-syntax 94
invalid-negated-syntax 94
invalid-short-equal-syntax 95
invalid-value 95
item 106

L

lispobj 107

M

missing-cmdline-argument 95

N

negatable 108
net.didierverna.clon 5, 27
net.didierverna.clon.core 7
net.didierverna.clon.setup 7, 39
net.didierverna.clon.setup/termio 6
net.didierverna.clon.termio 5

O

option 108
option-error 95

P

Package, net.didierverna.clon 27
Package, net.didierverna.clon.setup 39
path 109

S

sface 110
sheet 110
spurious-cmdline-argument 96
stropt 111

Structure, cmdline-option 97
Structure, frame 98
Structure, highlight-frame 98
Structure, highlight-property-instance 99
switch 111
switch-base 112
synopsis 113
System, net.didierverna.clon 5
System, net.didierverna.clon.core 7
System, net.didierverna.clon.setup 7
System, net.didierverna.clon.setup/termio 6
System, net.didierverna.clon.termio 5

T

text 114

U

unknown-cmdline-option-error 96
unrecognized-negated-call-error 97
unrecognized-short-call-error 97

V

valued-option 114

X

xswitch 116